# SAS® 9.4 Scalable Performance Data Engine: リファレンス (第 4 版)

SAS® ドキュメント

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2016. *SAS® 9.4 Scalable Performance Data Engine: リファレンス (第 4 版)*. Cary, NC: SAS Institute Inc.

**SAS® 9.4 Scalable Performance Data Engine: リファレンス (第 4 版)**



SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

November 2016



9.4-P1:engspde

# 目次

# 本書について

## SAS 言語の構文規則

### SAS 言語の構文規則の概要

SAS では、SAS 言語要素の構文ドキュメントに共通の規則を使用しています。これらの規則により、SAS 構文の構成要素を簡単に識別できます。規則は、次の項目に分類されます。

- 構文の構成要素
- スタイル規則
- 特殊文字
- SAS ライブラリと外部ファイルの参照

### 構文のコンポーネント

言語要素の多くでは、その構文の構成要素はキーワードと引数から構成されます。 キーワードのみ必要な言語要素もあります。また、キーワードに等号(=)が続く言語要素もあります。複数の引数を含む構文で区切り記号を使用する場合と使用しない場合を説明するために、引数の構文の形式が複数示されています。

キーワード
: プログラムの作成ときに使用する SAS 言語要素名です。 キーワードはリテラルであり、通常、構文の先頭の単語です。 CALL ルーチンでは、最初の 2 つの単語がキーワードです。

  これらの例の SAS 構文では、キーワードには太字が使用されています。

  **CHAR** (*string, position*)

  **CALL RANBIN** (*seed, n, p, x*);

  **ALTER** (*alter-password*)

  **BEST** *w.*

  **REMOVE** <*data-set-name*>

  この例では、CALL ルーチンの最初の 2 つの単語がキーワードです。

  **CALL RANBIN**(*seed*, *n*, *p*, *x*)

  引数なしで 1 つのキーワードから構成される SAS ステートメント構文もあります。

  **DO**;

*... SAS code ...*

**END;**

2 つのキーワード値のいずれか 1 つの指定が必要なシステムオプションもあります。

**DUPLEX** | **NODUPLEX**

プロシジャステートメントによっては、ステートメント構文中に複数のキーワードが含まれます。

**CREATE** <UNIQUE> **INDEX** *index-name* **ON** *table-name* (*column-1* <, *column-2*, ...>)

*引数*
数値定数、文字定数、変数、式のいずれかです。引数は、キーワードに続くか、キーワードの後ろの等号に続きます。SAS では、引数を使用して、言語要素を処理します。引数が必須の場合もオプションの場合もあります。構文では、オプションの引数は山かっこ( < > )で囲まれます。

この例では、*string* と *position* がキーワード CHAR に続きます。これらの引数は、CHAR 関数の必須引数です。

**CHAR** (*string, position*)

引数ごとに値が指定されます。この例の SAS コードでは、引数 *string* の値は'summer'、引数 *position* の値は 4 です。

```
x=char('summer', 4);
```

この例では、*string* および *substring* は必須引数ですが、*modifiers* と *startpos* はオプションです。

**FIND**(*string, substring* <,*modifiers*> <,*startpos*>

*argument(s)*
引数は必ず 1 つ必要であり、複数の引数が許可されます。引数の間はスペースで区切ります。カンマ( , )などの区切り記号は、引数間に必要ありません。

たとえば、MISSING ステートメントは、この形式で複数の引数を含みます。

**MISSING** *character(s)*;

<LITERAL_ARGUMENT> *argument-1* <<LITERAL_ARGUMENT> *argument-2* ... >
引数は必ず 1 つ必要であり、リテラル引数がこの引数に関連付けられます。リテラルと引数のペアは複数指定できます。リテラルと引数の間に区切り記号は必要ありません。省略記号(...)は、追加のリテラルと引数が許可されることを示します。

たとえば、BY ステートメントはこの引数を含みます。

**BY** <DESCENDING> *variable-1* <<DESCENDING> *variable-2* ...>;

*argument-1* <*option(s)*> <*argument-2* <*option(s)*> ...>
引数は必ず 1 つ必要であり、1 つ以上のオプションがこの引数に関連付けられます。複数の引数と関連するオプションを指定できます。引数とオプションの間に区切り記号は必要ありません。省略記号(...)は、追加の引数と関連するオプションが許可されることを示します。

たとえば、FORMAT プロシジャの PICTURE ステートメントは、この形式で複数の引数を含みます。

**PICTURE** name <(*format-option(s)*)>
<*value-range-set-1* <(*picture-1-option(s)*)>
<*value-range-set-2* <(*picture-2-option(s)*)> ...>>;

*argument-1=value-1 <argument-2=value-2 ...>*
引数には値を割り当てる必要があり、複数の引数を指定できます。省略記号(...)は、追加の引数が許可されることを示します。引数間に区切り記号は必要ありません。

たとえば、LABEL ステートメントは、この形式で複数の引数を含みます。

**LABEL** *variable-1=label-1 <variable-2=label-2 ...>*;

*argument-1 <, argument-2, ...>*
引数は必ず 1 つ必要であり、カンマまたは別の区切り記号で区切って複数の引数を指定できます。省略記号(...)は、カンマで区切られた引数が続くことを示します。SAS ドキュメントでは両方の形式が使用されます。

次に、この形式で指定された複数の引数の例を示します。

**AUTHPROVIDERDOMAIN** (*provider-1:domain-1 <, provider-2:domain-2, ...>*

**INTO** :*macro-variable-specification-1 <*, :*macro-variable-specification-2, ...>*

*注:* 通常、SAS ドキュメントのサンプルコードは、小文字の固定幅フォントを使用して表記されます。 コードの作成には、大文字も、小文字も、大文字と小文字の両方も使用できます。

## 書体に関する規則

SAS 構文の説明に使用されるスタイル規則には、大文字太字、大文字、斜体の規則も含まれます。

**大文字太字**
関数名やステートメント名などの SAS キーワードを示します。この例では、キーワード ERROR の表記には大文字太字が使用されています。

**ERROR** <*message*>;

大文字
リテラルの引数を示します。

この CMPMODEL=システムオプションの例では、BOTH、CATALOG、XML がリテラルです。

**CMPMODEL**=BOTH | CATALOG | XML |

*斜体*
ユーザー指定の引数または値を示します。斜体表記の項目は、ユーザー指定値であり、次のいずれかを表します。

- 非リテラル引数。この LINK ステートメントの例では、引数 *label* はユーザー指定値のため、斜体で表示されます。

  **LINK** *label*;

- 引数に割り当てられる非リテラル値。

  この FORMAT ステートメントの例では、引数 DEFAULT に変数の *default-format* が割り当てられます。

  **FORMAT** *variable(s)* <*format* > <DEFAULT = *default-format*>;

## 特殊文字

SAS 言語要素の構文には、次の特殊文字も使用されます。

=
等号は、一部の言語要素(システムオプションなど)のリテラル値を示します。

この MAPS システムオプションの例では、等号により MAPS の値が設定されます。

**MAPS** = *location-of-maps*

< >
山かっこはオプションの引数を示します。必須引数は山かっこで囲みません。

この CAT 関数の例では、少なくとも項目が 1 つ必要です。

**CAT** (*item-1* <, *item-2*, ...>)

|
縦棒は、値グループから 1 つの値を選択できることを示します。縦棒で区切られている値は、相互排他です。

この CMPMODEL=システムオプションの例では、引数を 1 つのみ選択できます。

**CMPMODEL**=BOTH | CATALOG | XML

...
省略記号は、引数の繰り返しが可能なことを示します。引数と省略記号が山かっこで囲まれている場合、その引数はオプションです。繰り返される引数には、その引数の前や後ろに、区切り記号を入れる必要があります。

この CAT 関数の例では、複数の *item* 引数が許可され、カンマで区切る必要があります。

**CAT** (*item-1 <, item-2*, ...>)

'*value*'または"*value*"
一重引用符や二重引用符付きの引数は、その値にも一重引用符または二重引用符を付ける必要があることを示します。

この FOOTNOTE ステートメントの例では、引数 *text* に引用符が付けられています。

**FOOTNOTE** <*n*> <*ods-format-options* '*text*' | "*text*">;

;
セミコロンは、ステートメントまたは CALL ルーチンの終わりを示します。

この例では、各ステートメントがセミコロンで終了しています。

```
data namegame;
length color name $8;
color = 'black';
name = 'jack';
game = trim(color) || name;
run;
```

## SAS ライブラリと外部ファイルの参照

多くの SAS ステートメントなどの言語要素では、SAS ライブラリと外部ファイルを参照します。 論理名(ライブラリ参照名またはファイル参照名)から参照を作成するのか、引用符付きの物理ファイル名を使用するかを選択できます。論理名を使用する場合、通常、参照の作成に SAS ステートメント(LIBNAME または FILENAME)を使用するのか、動作環境のコントロール言語を使用するのかを選択

します。複数の方法を使用して、SAS ライブラリと外部ファイルを参照できます。動作環境によっては使用できない方法があります。

SAS ドキュメントでは、外部ファイルを使用する例には斜体のフレーズ *file-specification* を使用します。 また、SAS ライブラリを使用する例には斜体フレーズ *SAS-library* を引用符で囲んで使用します。

infile *file-specification* obs = 100;
libname *libref* '*SAS-library*';

# SAS 9.4 Scalable Performance Data Engine の新機能

## 概要

9.4 では、次のオプションが追加または拡張されました。

- ALIGN=データセットオプション
- COMPRESS= LIBNAME ステートメントオプション
- ENCRYPT=AES (Advanced Encryption Standard)データセットオプション
- ENCRYPTKEY=データセットオプション
- IOBLOCKSIZE= LIBNAME ステートメントオプション
- SPDEFILECACHE システムオプション
- SAS 9.4 のメンテナンスリリース 1 では、SPD Engine で、DLDMGACTION=NOINDEX はサポートされていませんが、ABORT、FAIL、PROMPT、REPAIR はサポートされています。
- SAS 9.4 のメンテナンスリリース 3 では、“別のホスト上の SPD Engine ファイルへのアクセス” (3 ページ) という新しいセクションが追加されました。

## SPD Engine システムオプション

新しい SPD Engine システムオプションを使用すると次のことが行えます。

- SPDEFILECACHE システムオプションは、オープンされている SPD Engine データファイルのキャッシングを有効化または無効化します。詳細については、“SPDEFILECACHE システムオプション” (96 ページ)を参照してください。

## SPD Engine データセットオプション

新しい拡張された SPD Engine データセットオプションを使用すると次のことが行えます。

- 新しい ALIGN=データセットオプションは、変数アラインメントを指定します。詳細については、“ALIGN=データセットオプション” (50 ページ)を参照してください。
- ENCRYPT=データセットオプションは、より強力なセキュリティを実現するために、AES アルゴリズムをサポートするよう拡張されました。詳細については、“SPD Engine データセットの暗号化” (21 ページ)を参照してください。
- 新しい ENCRYPTKEY=データセットオプションは、AES 暗号化で使用するキー値を指定します。詳細については、“ENCRYPTKEY=データセットオプション” (65 ページ)を参照してください。

## SPD Engine LIBNAME ステートメントオプション

新しい LIBNAME ステートメントオプションを使用すると、次の処理を行えます。

- SAS 9.4 のメンテナンスリリース 2 では、新しい IOBLOCKSIZE= LIBNAME ステートメントオプションを使用すると、I/O 操作で使用されるオブザベーションブロックのバイト数でサイズを指定できます。詳細については、“IOBLOCKSIZE= LIBNAME ステートメントオプション” (39 ページ)を参照してください。
- SAS 9.4 のメンテナンスリリース 2 では、COMPRESS= LIBNAME ステートメントオプションを使用すると、SPD Engine データセットを作成時にディスク上で圧縮できます。

1 章
# 概要: SPD Engine

# SPD Engine について

SPD Engine は、ハイパフォーマンスデータ配信向けに設計されています。これにより、アプリケーションで、処理する SAS データに迅速にアクセスできるようになります。SPD Engine は、データを合理化されたファイル形式に編成し、複数の CPU を利用して並列入力機能を実行することにより、データをアプリケーションに迅速に提供します。

SPD Engine では、*スレッド*を使用して、非常に迅速に並列処理でデータブロックが読み取られます。タスクはオペレーティングシステムに関連して実行されます。このため、コンピュータの任意の使用可能 CPU でスレッドを実行できます。スレッド化された読み取りタスクは SPD Engine 機能の重要な部分ですが、SPD Engine の真価は、SAS データの構造化方法にあります。SPD Engine では、データがファイル形式に編成されます。これにはデータの分割も含まれます。このデータ構造では、スレッド(並列実行)を許可することにより、読み取りタスクを効率的に実行できます。

SPD Engine は、非常に大きなデータセットの処理では、デフォルト Base SAS Engine の高速の代替方法となります。同エンジンは、数 10 億件のオブザベーションを含んでいるデータセットを読み取ります。たとえば、これには、一部のオペレーティングシステムで課されるサイズ制限を超えて拡張するデータセット、ならびに SAS 分析ソフトウェアおよびプロシジャでより迅速な処理が必要なデータセットが含まれます。

SPD Engine は、次の方法でパフォーマンスを向上させます。

- 数百 GB のデータのサポート
- 対称型マルチプロセッサ(SMP)コンピュータと超並列プロセッサ(MPP)コンピュータでのスケーラビリティ
- 並列処理での WHERE 選択
- 並列処理でのロード
- 並列処理でのインデックス作成
- 並列処理でのデータのアプリケーションへの配信
- BY ステートメントでの自動並べ替え

SPD Engine は、UNIX、Windows、および z/OS (zFS ファイルシステムのみ)で実行されます。

*注:* スケーラビリティの詳細については、Scalability and Performance Community フォーカス領域(http://support.sas.com/rnd/scalability)を参照してください。システム要件については、Install Center (http://support.sas.com/documentation/installcenter)を参照してください。

# SPD Engine 互換性

## SAS 9 のアップグレード

SAS 9 の前のリリースからアップグレードする場合、動作環境が同一のままであれば、データセットを移行する必要はありません。複数のホストにわたるアップ

グレードの場合(例:32 ビットから 64 ビットの Windows 動作環境へ)、データセットを移行する必要があります。データセットを移行するには、CIMPORT、COPY または CPORT を使用します。詳細については、*Base SAS Procedures Guide* を参照してください。

*注:* MIGRATE プロシジャでは、SPD Engine はサポートされていません。

### 別のホスト上の SPD Engine ファイルへのアクセス

SPD Engine は、データセットを複数のホストファミリ間で共有できませんが、1 つのホストファミリ内で共有できます。たとえば、両方のホストのデータ表現が同じである限り、別のホスト上の SPD Engine ファイルにアクセス(読み取りおよび更新)できます。

SPD Engine は次のホスト上で動作します。

| | |
|---|---|
| Itanium 上の HP-UX IPF、または HP-UX | SPARC 上の Solaris 64 ビットまたは Solaris |
| x64(x86-64)上の LINUX | x64(x86-64)上の Solaris 10 |
| MVS_32 | Windows 32 ビット |
| Power 上の AIX 64 ビットまたは AIX | Windows 64 ビット |

データ表現に基づいて、データセットは次のホストファミリに分類されます。

- 次のホスト上で動作するビッグエンディアン:
  - Itanium 上の HP-UX IPF、または HP-UX
  - Power 上の AIX 64 ビットまたは AIX
  - SPARC 上の Solaris 64 ビットまたは Solaris
- 次のホスト上で動作するビッグエンディアン:
  - x64(x86-64)上の LINUX
  - x64(x86-64)上の Solaris 10
- z/OS 32 ビット上の z/OS
- Windows 32 ビット上の Windows
- Windows 64 ビット上の Windows x64

## SMP コンピュータの使用

SPD Engine では、対称型マルチプロセッシングと呼ばれるハードウェアとソフトウェアのアーキテクチャが活用されます。SMP コンピュータには、複数の中央処理装置(CPU)、およびスレッドをサポートするオペレーティングシステムがあります。SMP コンピュータは通常、複数のコントローラ、およびコントローラごとの複数のディスクドライブで構成されます。SPD Engine では、データファイルの読み取り時に、各 CPU に対して 1 つ以上のスレッドが起動されます。これらのスレッドは、CPU ごとに 1 つ以上のコントローラによって駆動され、複数のディスクドライブから並列処理でデータを読み取ります。SMP コンピュータで実行する SPD Engine では、所定の経過時間内ではるかに多くのデータを読み取り、アプリケーションに配信する機能が提供されます。

5 つの CPU と 10 のディスクドライブを備えた SMP コンピュータでデータセットを読み取ると、シングル CPU コンピュータの I/O よりも 5 倍速く処理することが可能になります。スレッド I/O に加えて、SMP コンピュータでは、アプリケーションプロセスのスレッド化が可能になります(たとえば、SAS 9.1 以降の SORT プロシジャでのスレッド化された並べ替えなど)。

SMP コンピュータの正確な CPU 数は、製造元とモデルによって変わります。コンピュータのオペレーティングシステムも専門化され、並列実行のためにコードセグメントをスケジューリングする機能が必要とされます。オペレーティングシステムカーネルがスレッド化されると、実行スレッド間の競合が防止されるため、パフォーマンスがさらに向上します。

スレッドオペレーティングシステムで管理された SMP コンピュータ上でスレッドを実行すると、使用可能な CPU が同時に作動します。CPU とスレッドの相乗効果によって、ソフトウェアで、処理パフォーマンスのスケール変更が可能になります。その結果、スケーラビリティによって、データセットの作成、データの追加、および WHERE ステートメントを使用したデータのクエリなどのタスクの全体処理速度が大幅に増加します。

# SPD Engine を使用した SAS データの編成

## SPD Engine の SAS データの編成方法

SPD Engine でハイパフォーマンス処理のためにデータが編成されるので、SPD Engine データセットはデフォルト Base SAS Engine データセットと物理的に異なります。デフォルト Base SAS Engine では、ファイルのデータとデータディスクリプタ(メタデータ)の両方が含まれる単一データファイルにデータが格納されます。SPD Engine では、データとデータディスクリプタに対して別々のファイルが作成されます。さらに、データセットにインデックスが付いている場合、SPD Engine で、インデックスごとに 2 つずつインデックスファイルが作成されます。これら 4 種類のファイルのそれぞれが SPD Engine *コンポーネントファイル*と呼ばれ、それぞれの識別ファイル拡張子が存在します。

メタデータは単独の物理ファイルですが、複数の物理ファイルを占有する場合もあり、各ファイルはファイル名に.mdf を持ちます。データコンポーネントは一つ以上の物理ファイルで、各ファイルはファイル名に.dpf を持ちます。インデックスが定義されるとインデックスコンポーネントが作成され、各インデックスは 2 つの物理ファイルを持ちます。

- ファイル名に.hbx を持つファイル
- ファイル名に.idx を持つファイル

これらのコンポーネントファイルのそれぞれを 1 つ以上の物理ファイルで構成し、コンポーネントが複数のボリュームにわたっていても 1 つの論理ファイルとして参照するようにできます。たとえば、SPD Engine では、データを含む物理ファイルを多数作成できますが、データを含むファイルを、SPD Engine データセットの単一データコンポーネントとして参照できます。

*メタデータ*コンポーネントと *インデックス*コンポーネントは、次の 2 点が *データ*コンポーネントと異なります。

1. PARTSIZE=オプションを使用してデータコンポーネントファイルの固定長パーティションサイズを指定できます。メタデータコンポーネントまたはインデックスコンポーネントのパーティションサイズは指定できません。

2. データコンポーネントファイルは、すべての定義パスにわたって周期的に作成されます。メタデータコンポーネントファイルとインデックスコンポーネントファイルは、シングル定義パスで、そのパスがいっぱいになるまで作成され、その後で次の定義パスが使用されます。

### メタデータコンポーネントファイル

SPD Engine データセットでは、説明的なメタデータが、ファイル拡張子.mdf のファイルに格納されます。通常、SPD Engine データセットの.mdf ファイルは 1 つだけです。その.mdf ファイルには、その他のコンポーネントファイルすべてのパス名が含まれます。

### インデックスコンポーネントファイル

ファイルにインデックスが付いている場合、SPD Engine で、インデックスごとに 2 つずつインデックスファイルが作成されます。これらの各ファイルにインデックスの特定ビューが含まれます。

- .hbx ファイル拡張子の付いたインデックスファイルには、グローバルインデックスが含まれます。
- .idx ファイル拡張子の付いたインデックスファイルには、セグメントインデックスが含まれます。

### データコンポーネントファイル

SPD Engine データセットのデータコンポーネントでは、パスまたはデバイスごとにファイル(パーティション)が 1 つだけではなく複数存在する可能性があります。これらの各パーティションは固定長で、その長さは SPD Engine データセットの作成時にユーザーによって指定されます。

データコンポーネントファイルのサイズ指定によって、アプリケーションのパフォーマンスを調整できます。パーティションとはスレッド化可能単位です。すなわち、各パーティション(ファイル)は 1 スレッドで読み取られます。2 章, “SPD Engine ファイルの作成とロード” (11 ページ) では、SPD Engine でのデータ、メタデータおよびインデックスの格納に関する詳細を提供します。

## デフォルト Base SAS Engine と SPD Engine の比較

### 比較の概要

デフォルト Base SAS Engine データセットと SPD Engine データセットには多くの類似点があります。両方とも、ファイルの論理コレクションである SAS ライブラリにデータを格納します。SPD Engine データライブラリは複数のデバイスやファイルシステムにまたがるので、SPD Engine は非常に大きなデータセットの使用に最適です。また、SPD Engine では、LIBNAME ステートメントでコンポーネントごとに別々のディレクトリ(またはデバイス)を指定できます。2 章, “SPD Engine ファイルの作成とロード” (11 ページ) では、SPD Engine データライブラリの設計および設定に関する詳細を提供します。

## SPD Engine ライブラリとファイルシステム

SPD Engine ライブラリには、データファイル、メタデータファイルおよびインデックスファイルを含められます。SPD Engine では、カタログ、SAS ビュー、MDDB、その他のユーティリティ(バイト)ファイルはサポートされません。

SPD Engine では、z/OS の zFS ファイルシステムが使用されます。

## ユーティリティファイルワークスペース

ユーティリティファイルは、追加領域を必要とする SPD Engine 操作時(並列インデックスの作成時や、非常に大きなファイルの並べ替え時など)に生成されます。すべてのプラットフォームにデフォルトの場所が存在しますが、大量の処理データがある場合、デフォルトの場所の容量が不十分になる可能性があります。SPD Engine システムオプション SPDEUTILLOC=では、ユーティリティスクラッチファイルを格納する一連のファイル場所を指定できます。詳細については、“SPDEUTILLOC=システムオプション” (99 ページ)を参照してください。

## 一時データセットの格納

中間データセットを格納するライブラリを作成するには、LIBNAME ステートメントに SPD Engine オプション TEMP=を指定します。現在のアプリケーションで 1 レベル名を使用してこれらの中間ファイルを参照する場合は、USER=システムオプションでライブラリを指定します。

次のコード例では、中間データセットのユーザーライブラリ参照名が作成されます。これはセッションの終了時に削除されます。

```
libname user spde 'SAS-library' temp=yes;
data a; x=1;
run;
proc print data=a;
```

構成ファイルで USER=オプションを設定し、1 レベル名で中間データセットを参照するアプリケーションを SPD Engine で実行するようにできます。

## デフォルト Base SAS Engine データセットと SPD Engine データセットの相違

次の表は、SPD Engine 機能とデフォルト Base SAS Engine 機能の比較です。

**表 1.1** デフォルト Base SAS Engine データセットと SPD Engine データセットの比較

| 機能 | SPD Engine | デフォルト Base SAS Engine |
|---|---|---|
| 分割データセット | あり | なし |
| 並列処理での WHERE 最適化 | あり | なし |
| 最下位ロックレベル | メンバ | レコード |
| 指定データセットでの複数 SAS セッションからの同時アクセス | READ (INPUT オープンモード) | READ と WRITE (すべてオープンモード) |

| 機能 | SPD Engine | デフォルト Base SAS Engine |
|---|---|---|
| SAS/CONNECT 経由のリモートコンピューティング | なし | あり |
| SAS/CONNECT 経由のデータ転送 | なし | あり |
| SAS/CONNECT 経由の RLS (リモートライブラリサービス) | なし | あり |
| SAS/CONNECT 経由の使用可能性 | なし | あり |
| SAS/SHARE でのサポート | なし | あり |
| SAS BY 処理のための自動並べ替え (BY 処理をサポートするためにデータの一時コピーを並べ替える) | あり | なし |
| ユーザー定義の出力形式と入力形式 | あり(WHERE 以外)* | あり |
| カタログ | なし | あり |
| ビュー | なし | あり |
| MDDB | なし | あり |
| 一貫性制約 | なし | あり |
| データセット世代 | なし | あり |
| CEDA | なし | あり |
| 監査証跡 | なし | あり |
| NLS トランスコーディング | なし | あり |
| COMPRESS= | YES\|NO\|CHAR\|BINARY (ファイルが暗号化されていない場合のみ) | YES\|NO\|CHAR\|BINARY |
| DLCREATEDIR | なし | あり |
| ENCRYPT= | COMPRESS=と一緒には使用不可 | COMPRESS=と一緒に使用可能 |
| 暗号化 | あり | あり |
| AES 暗号化 | データファイルおよびインデックスファイルのみ | あり |
| FIRSTOBS=システムオプションおよびデータセットオプション | なし | あり |

| 機能 | SPD Engine | デフォルト Base SAS Engine |
|---|---|---|
| OBS=システムオプションおよびデータセットオプション | あり(ENDOBS=または STARTOBS= SPD Engine オプションなしで使用する場合) | あり |
| EXTENDOBSCOUNTER=システムオプションおよびデータセットオプション | なし | あり |
| 拡張属性 | なし | あり |
| 関数とコールルーチン | あり(ただし一部例外あり) | あり |
| テーブルを OS ユーティリティによって異なるディレクトリまたはフォルダに移動 | なし | あり |
| 物理的な順序で返されるオブザベーション | なし(BY または WHERE が存在する場合) | あり |
| DLDMGACTION=システムオプションおよびデータセットオプション | ABORT \| FAIL\| PROMPT \| REPAIR はあり、NOINDEX はなし | あり |

* WHERE 処理では、関数とコールルーチンはスーパーバイザに渡されて処理されます。したがって、並列では処理されません。

## デフォルト Base SAS Engine データセットと SPD Engine データセットの相互運用性

デフォルト Base SAS Engine データセットは、SPD Engine でアクセスできるように、SPD Engine 形式に変換する必要があります。デフォルト Base SAS Engine データセットは、COPY プロシジャ、APPEND プロシジャまたは DATA ステップを使用すると容易に変換できます。(PROC MIGRATE は使用できません。)さらに、既存 SAS プログラムの大部分は、LIBNAME ステートメント以外、ほとんど修正せずに、SPD Engine ファイルに対して実行できます。 2 章, “SPD Engine ファイルの作成とロード” (11 ページ) では、デフォルト Base SAS Engine データセットから SPD Engine 形式への変換に関する詳細を提供します。

## SPD Engine ファイルの共有

SPD Engine では、メンバレベルのロックがサポートされます。つまり、複数のユーザーが INPUT で同じ SPD Engine データセットを開くことができます(読み取り専用)。ただし、更新またはインデックス作成のために SPD Engine データセットを開いた場合は、そのユーザーのみアクセス可能になります。

# I/O のパフォーマンスを向上させる機能

## I/O パフォーマンス向上の概要

SPD Engine には、I/O パフォーマンスを向上させる機能がいくつかあります。これらの機能によって I/O バウンドアプリケーションのパフォーマンスを劇的に高められます。このとき、大量のデータはアプリケーションに配信して処理する必要があります。

## 複数のディレクトリパス

SPD Engine ではいくつかのボリュームにわたる複数の物理ファイルを単一の論理ファイルとして参照できるので、コンポーネントの種類ごとに複数のディレクトリパスとデバイスを指定できます。非常に大きなデータセットの場合、この機能によって、オペレーティングシステムで課される可能性があるファイルサイズ制限をすべて回避できます。

## データファイルと関連インデックスの物理的分離

コンポーネントファイルの各種類はそれぞれ異なる場所に格納できるので、コンポーネントファイルの格納場所の決定時にファイル依存関係を考慮する必要はありません。考慮が必要なのは、コスト、パフォーマンス、およびディスク領域の可用性のみです。

## WHERE の最適化

SPD Engine では、WHERE ステートメントで指定された基準を満たすためにオブザベーションを評価するのに最適な方法が自動的に決定されます。WHERE ステートメントの効率性は、式の変数にインデックスを付けるかどうかなどの因子によって変わります。WHERE 評価プランナが SPD Engine に含まれます。これにより、インデックスを使用する WHER 式評価の最適化に使用する最適な方法を選択できます。

# 処理パフォーマンスを向上させる機能

## 自動的な並べ替え機能

SPD Engine の自動並べ替え機能によって、大きなデータセットを処理する SAS アプリケーションの時間とリソースが節約されます。SPD Engine では、BY 句を指定した SAS ステートメントをサブミットする前に SORT プロシジャを呼び出す必要はありません。SPD Engine で BY 句が検出される際、データはまだ並べ替えられておらず、BY 変数のインデックスも付いていません。SPD Engine による自動的なデータ並べ替えでは、永久データセットに影響は及ばず、新しい出力データセットも作成されません。

### インデックスを使用したクエリ

パフォーマンスを最大にするには大きなデータセットにインデックスを付けます。インデックスによって、インデックス付き変数に対する迅速な WHERE 式評価が可能になります。SPD Engine は、複数の CPU を利用して、インデックスコンポーネントファイルを効率的に検索します。

*注:* 変数名に次の特殊文字のいずれかが含まれている場合は、(VALIDMEMNAME=EXTEND オプションを指定しても)変数に対して単一インデックスも複合インデックスも作成できません。

" * | \ : / < > ? - .

### 並列処理でのインデックス作成

さらに、SPD Engine では、大きなデータセットのインデックス付けに時間がかからないように、並列インデックス作成がサポートされています。SPD Engine では、データセットの追加操作または挿入操作が、並列実行の可能な一連のステップに分解されます。並列処理のレベルは、データセットに存在するインデックスの数によって決まります。インデックスが多いほど、インデックス作成時に並列処理が多く活用されます。ただし、インデックスの作成には、ユーティリティのファイル領域とメモリリソースが必要です。

*注:* 変数名に次の特殊文字のいずれかが含まれている場合は、(VALIDMEMNAME=EXTEND オプションを指定しても)変数に対して単一インデックスも複合インデックスも作成できません。

" * | \ : / < > ? - .

## SPD Engine のオプション

SPD Engine では、多くのデフォルト Base SAS Engine オプションが取り扱われます。それに加えて、SPD Engine でのみ使用されるオプションもあります。そのオプションを使用すると、SPD Engine のライブラリと処理の管理が促進されます。参照:

- SPD Engine データセットオプション (47 ページ)
- SPD Engine LIBNAME ステートメントオプション (27 ページ)
- SPD Engine システムオプション (89 ページ)

2 章
# SPD Engine ファイルの作成とロード

# SPD Engine ファイルの作成とロードについて

このセクションでは、SPD Engine ライブラリの割り当て、ならびに SPD Engine のデータとインデックスの作成およびロードについて詳細を提供します。また、これらのタスクに関連するパフォーマンスの考慮事項についても説明します。

# ライブラリ領域の割り当て

## ライブラリ領域の割り当て方法

SPD Engine の分割データ読み取りおよびスレッド機能によるパフォーマンスの向上を実現するには、SPD Engine ライブラリを正しく構成および管理する必要があります。SAS System 管理者はこれらのタスクを最適な形で実行します。

SPD Engine データセットでは、各種コンポーネントの格納に十分な領域を有するファイルシステムが必要です。多くの場合、そのファイルシステムには、それらのコンポーネントに対する複数のディレクトリが含まれます。通常、シングルディレクトリパス(ファイルシステムの一部)は、ファイルシステム全体のボリューム制限に制約されます。この制限とは、ファイルシステム用に構成されたディスク領域の最大容量です。

この最大ディスク領域の範囲内で、SPD Engine コンポーネントファイルすべてにとって十分な領域を割り当てる必要があります。各ライブラリに必要な記憶容量を見積もるには、各コンポーネントファイルの処理方法を理解しておくことが重要です。

## シングルパスのすべてのコンポーネントの領域の構成

最も単純な SPD Engine ライブラリ構成では、すべての SPD Engine コンポーネントファイル(データファイル、メタデータファイルおよびインデックスファイル)が、*プライマリパス*というシングルパスに存在します。プライマリパスは、LIBNAME ステートメントのデフォルトパス指定です。次の LIBNAME ステートメントでは、MyLib ライブラリに対してプライマリファイルシステムが設定されます。

libname mylib spde '*/disk1/spdedata*';

他のパスオプションが指定されていないため、すべてのコンポーネントファイルがこのプライマリパスに作成されます。すべての種類のコンポーネントファイルをプライマリパスに格納するのは、簡単で、非常に小容量のデータセットでは効果的です。ただし、コンポーネントを別々に格納することによって達成可能なパフォーマンスの向上や、複数の CPU という利点は得られません。

*注:* SPDEngine では、完全パス名を指定する必要があります。

## コンポーネントファイルの種類別のライブラリ領域の構成

ほとんどのサイトでは、SPD Engine は非常に大容量のデータの管理に使用されます。データには何千もの変数を含められ、その一部にはインデックスを付けられます。それらのサイトでは通常、各種コンポーネントに対して別々の記憶域パ

スが定義されます。さらに、ディスクストライピングおよび RAID (Redundant Array of Independent Disks)を使用すると、非常に効率的になる可能性があります。詳細については、Scalability and Performance の“SPD Engine Disk I/O Setup” (http://support.sas.com/rnd/scalability/spde/setup.html)を参照してください。

ライブラリのすべてのデータセットに対するメタデータコンポーネントファイルは、プライマリパスに存在する必要があります。

さらに、データコンポーネントファイルとインデックスコンポーネントファイルに別々のパスを指定すると、パフォーマンスが向上します。別々のパスを指定するのは、読み取り負荷を複数のディスクドライブに分散するためです。データコンポーネントファイルとインデックスコンポーネントファイルを分けておくと、複雑な WHERE 評価では特に、ディスクの競合を防ぎ、達成可能な*並列処理*のレベルを上げるのに役立ちます。次のコード例では、メタデータコンポーネントファイルのプライマリパスが指定されます。このコードでは、DATAPATH=オプション (34 ページ) および INDEXPATH=オプション (38 ページ) を使用して、データコンポーネントファイルとインデックスコンポーネントファイルに対して別々のパスが追加指定されます。

```
libname all_users spde '/disk1/metadata'
  datapath= ('/disk2/userdata' '/disk3/userdata')
  indexpath= ('/disk4/userindexes' '/disk5/userindexes');
```

メタデータコンポーネントファイルは、プライマリパスの disk1 に格納されます。データコンポーネントファイルは disk2 および disk3、インデックスコンポーネントファイルは disk4 および disk5 です。すべてのパス指定において、完全パス名を指定する必要があります。

**注意:**

**プライマリパスはライブラリごとに一意であることが必要です。** プライマリパスが同じで、その他のパスに相違があるライブラリ参照名を 2 つ作成した場合、データが失われる可能性があります。NFS はプライマリパス以外のパスでは使用できません。

*注:* ローカルでマウントされたドライブにデータを格納し、リモートコンピュータからデータにアクセスする場合は、LIBNAME の指定時にリモートパス名を使用します。**/data01** および**/data02** が localA コンピュータにローカルでマウントされたドライブである場合は、LIBNAME ステートメントでパス名**/nfs/localA/data01** および**/nfs/localA/data02** を使用します。

## コンポーネントファイル別の領域の予想

SPD Engine ライブラリ領域を正しく構成するには、SPD Engine コンポーネントファイルの相対サイズについて理解する必要があります。次の情報は概要を示しています。詳細については、Scalability and Performance の“SPD Engine Disk I/O Setup” (http://support.sas.com/rnd/scalability/spde/setup.html)を参照してください。

メタデータコンポーネントファイルは相対的に小さいファイルですが、指定するプライマリパスは、ライブラリのメタデータコンポーネントファイルをすべて含めるのに十分な大きさにする必要があります。メタデータコンポーネントファイルは、パスで使用可能な領域よりも大きくすることはできません。

インデックスコンポーネントファイル(.idx と.hbx の両方)は、各インデックスの個別の値数、およびインデックスが単一インデックスか複合インデックスかに応じて、中容量から大容量になります。インデックスコンポーネントファイルが、現在のファイルパスで使用可能領域よりも大きくなると、次のパスに新しいインデックスコンポーネントファイルが作成されます。

データコンポーネントファイルは、データ容量と、データセットに指定するパーティションサイズによっては、多数に及ぶ可能性があります。各データパーティションが別々のデータコンポーネントファイルとして格納されます。データパーティションのサイズは、 PARTSIZE= LIBNAME ステートメントオプション (42 ページ)または PARTSIZE=データセットオプション (78 ページ)で指定されます。パーティションサイズを指定できるコンポーネントファイルはデータコンポーネントファイルのみです。

## メタデータコンポーネントファイルの記憶域

### メタデータコンポーネントファイル

SPD Engine データセットのメタデータコンポーネントファイルには、データセットについての説明的な情報、ならびにその構成データコンポーネントファイルおよびインデックスコンポーネントファイルへのパス名が格納されます。この概念を理解しておくことは非常に重要です。これはデータセット(および関連するメタデータコンポーネントファイル)をライブラリに追加可能かどうかに直接影響するからです。

ライブラリのすべてのデータセットに対するメタデータコンポーネントファイルは、プライマリパスでの指定と同じ場所に存在する必要があります。実質的に、プライマリパスのファイルは、ライブラリ全体に対してディレクトリのように動作します。SPD Engine データセットにアクセスがあると、SPD Engine はまずデータセットのメタデータコンポーネントファイルを開いて、その属性を決定し、その他のコンポーネントファイルすべてにアクセスできるかどうかを決定します。ライブラリに対して新しいデータセットが作成されるときに、プライマリパス内の領域がいっぱいになっている場合、SPD Engine はそのパスでのメタデータコンポーネントファイルの作成を開始できず、適切なエラーメッセージが出て作成操作が失敗します。この場合、新しいデータセットを正常に作成するには、プライマリパスの領域を解放するか、または新しいライブラリを割り当てて、その新しいライブラリに一部またはすべてのデータセットをコピーする必要があります。データコンポーネントファイルとインデックスコンポーネントファイルには、その制限はありません。データコンポーネントファイルとインデックスコンポーネントファイルには後から追加領域を指定できます。

特定のアクションが原因となって、メタデータコンポーネントファイルが拡張され、ファイルサイズや領域の制限を超過することがあります。その場合、SPD Engine では、オーバーフローに合わせて、メタデータコンポーネントファイルのパーティションがもう 1 つ作成されます。新しいメタデータパーティションは、プライマリパスか、 “METAPATH= LIBNAME ステートメントオプション” (41 ページ)で指定されたパスに存在します。METAPATH=オプションでは、新しいデータセットの最初のメタデータパーティションの領域を作成することはできません。METAPATH=オプションでは、最初の領域を超えたメタデータコンポーネントファイルにのみ領域が指定されます。

### インデックスコンポーネントファイルの記憶域

インデックスコンポーネントファイルは、オーバーフロースペースに格納されます。インデックスコンポーネントファイルが大きくなってファイルサイズや領域制限を超えると、SPD Engine では、オーバーフローに合わせて、インデックスコンポーネントファイルのパーティションがもう 1 つ作成されます。INDEXPATH=オプションで複数のファイルパスが指定された場合、最初の使用可能領域でインデックスコンポーネントファイルが作成されて、前のパスがいっぱいになると次のパスにオーバーフローします。メタデータコンポーネントファイルとは違って、インデックスコンポーネントファイルはプライマリパスに置く必要はありません。

### データコンポーネントファイルの記憶域

パーティションサイズを指定できるファイルはデータコンポーネントファイルのみです。分割データはスレッドで容易に処理できるので、コンピュータの複数 CPU が最大限に活用されます。データコンポーネントファイルのパーティションサイズは固定されています。これはデータセットの作成時に設定されます。デフォルトは 128MB ですが、PARTSIZE=オプションを使用して異なるパーティションサイズを指定できます。パフォーマンスは適切なパーティションサイズによって決まります。このためデータのサイズと用途を理解しておく必要があります。バランスの取れたオブザベーション数をもたらすパーティションサイズで SPD Engine データセットを作成できます。詳細については、“PARTSIZE= データセットオプション” (78 ページ) および“PARTSIZE= LIBNAME ステートメントオプション” (42 ページ)を参照してください。指定データセットの各データパスで多くのデータパーティションを作成できます。SPD Engine では、DATAPATH=オプションで指定したファイルパスを使用して、パーティションが周期的に分散されます。SPD Engine では、指定パスの 1 つに最初のデータパーティション、次のパスに 2 番目のパーティションというように、順番に作成されます。SPD Engine では、データセットのすべてのデータパーティションが格納されるまで、必要な回数だけ、ファイルパスで処理が循環します。最初のパーティションに対するファイルパスは、ランダムに選択されます。LIBNAME ステートメントで次のように指定したとします。

```
datapath=('/data1' '/data2')
```

SPD Engine では、/DATA1 に最初のパーティション、/DATA2 に 2 番目のパーティション、/DATA1 に 3 番目のパーティション、というように格納されていきます。データパーティションの周期的分散によってディスクストライピングが行われます。これにより非常に効率がよくなる可能性があります。ディスクストライピングの詳細については、Scalability and Performance の“SPD Engine Disk I/O Setup” (http://support.sas.com/rnd/scalability/spde/setup.html)を参照してください。

### パスの初期セット

次の例では、LIBNAME ステートメントで、プライマリパスとして MYLIB ディレクトリが指定されます。このパスはメタデータパーティションの格納に使用されます。データパーティションおよびインデックスパーティションの格納用に他のデバイスおよびディレクトリが指定されます。

```
libname myref spde 'Mylib'
  datapath=('/mydisk30' '/mydisk31')
  indexpath=('/mydisk36');
```

MyLib ライブラリで作成されたデータセットがすべて、複数のデータパーティションを含めるのに十分な大きさがあると仮定すると、そのすべてのメタデータは MyLib に格納されます。データは**/mydisk30** および**/mydisk31**、インデックスは**/mydisk36** に置かれます。具体的には、それらのデータセットのメタデータコンポーネントファイルにそれらのパス名が含まれます。

### 後続パスの追加

後からさらに領域が必要になった場合(データをさらに追加する場合など)、次の例のように、追加デバイスをデータパーティションとインデックスパーティションに追加できます。

```
libname myref spde 'Mylib'
  datapath=('/mydisk30' '/mydisk31' '/mydisk32')
  indexpath=('/mydisk36' '/mydisk37');
```

MyLib ライブラリで作成されたすべてのデータセットでは、メタデータは MyLib に、データは 3 つのうち 1 つ以上のパスに、インデックスはいずれも**/mydisk36**

または**/mydisk37** に格納されます。データが既存データセットに追加された場合、新しいデータは 3 つのうち 1 つ以上のパスに入れられ、それに応じてメタデータコンポーネントファイルが更新されます。

1 つ以上のデータパーティションまたはインデックスパーティションに空き領域があまりない場合は、次回の指定時に LIBNAME ステートメントで除外できます。

```
libname myref spde 'Mylib'
  datapath=('/mydisk31' '/mydisk32' '/mydisk33')
  indexpath=('/mydisk37' '/mydisk38');
```

データセットのメタデータには使用パスがすべて含まれるため、SPD Engine では、除外されたパスを使用するデータセットでもアクセスできます。

#### パスの省略

ライブラリ内のデータセットのみ読み取る必要がある場合、必要なパス情報はすべてメタデータコンポーネントファイルにすでに存在するので、追加の DATAPATH=および INDEXPATH=オプションなしで LIBNAME ステートメントを指定できます。

```
libname myref spde 'Mylib';
```

### コンポーネントファイルの名前変更、コピー、移動

**注意:**

**SPD Engine のデータセットやそのコンポーネントファイルの名前変更、コピー、移動にはオペレーティングシステムコマンドを使用しないでください。**

ある場所から別の場所に SPD Engine データセットをコピーするには COPY プロシジャ、SPD Engine データセットの名前変更や削除には DATASETS プロシジャを常に使用してください。

## ディスクストライピングと大容量ディスクアレイを使用した場合の効率

システム内にあるファイル作成ユーティリティで、ファイルシステムの制限を上書きし、単一のディスク上の領域より大きいファイルシステム(ボリューム)を作成できる場合があります。RAID などの複数のディスクデバイスにまたがる SPD Engine ライブラリを割り当てるには、このユーティリティを使用します。RAID 構成では、ディスクストライピングという手法によって、I/O が大幅に拡張されます。ディスクストライピングおよび RAID の詳細については、Scalability and Performance の“SPD Engine Disk I/O Setup” (http://support.sas.com/rnd/scalability/spde/setup.html)を参照してください。

*注:* 保存のために Hadoop 分散ファイルシステム(HDFS)を使用する場合は、*SAS SPD Engine: Storing Data in the Hadoop Distributed File System* を参照してください。

# デフォルト Base SAS Engine データセットから SPD Engine データセットへの変換

## COPY プロシジャと APPEND プロシジャの使用

既存のデフォルト Base SAS Engine データセットを SPD Engine データセットに変換するには、次のメソッドを使用します。

- PROC COPY
- PROC APPEND

いくつかの制限が適用されます。デフォルト Base SAS Engine データセットに一貫性制約がある場合は、データセットが SPD Engine 形式で作成されるときに、その一貫性制約は解除されます。次のファイル特性表は、特性の保持や解除が行われる可能性があるかどうか、または変換によって結果的にエラーが発生するかどうかを示しています。

**表 2.1** Base SAS Engine データセット特性の変換結果

| Base SAS Engine データセット特性 | 変換結果 |
|---|---|
| インデックス | SPD Engine で再作成<br>(ASYNCINDEX=YES の場合は並列処理) |
| デフォルト Base SAS Engine COMPRESS=YES \|CHAR \|BINARY * | データセットが暗号化されていない場合、圧縮ありの変換 |
| デフォルト Base SAS Engine ENCRYPT=YES * | 暗号化ありの変換 |
| 一貫性制約 | エラーなしで解除 |
| 監査ファイル | 警告付き解除 |
| 世代ファイル | 警告付き解除 |
| 拡張属性 | 警告付き解除 |

* デフォルト Base SAS Engine データセットで圧縮と暗号化の両方が行われている場合、圧縮は解除されますが、暗号化は保持されます。SAS では、圧縮のかわりにデータセットのセキュリティが保持されます。

## PROC COPY を使用したデフォルト Base SAS Engine データセットの変換

既存のデフォルト Base SAS Engine データセットから SPD Engine データセットを作成するには、単に COPY プロシジャを使用します。PROC COPY ステートメントでは、デフォルト Base SAS Engine 形式のデータセット Local.Racquets が、新しい SPD Engine 形式のデータセット Sport.Racquets にコピーされます。

```
libname sport spde 'conversion_area';

proc copy in=local out=sport;
  select racquets;
run;
```

デフォルト Base SAS Engine データセットのインデックスが SPD Engine インデックスとして自動的に再生成される場合でも(.hdx ファイルと.idx ファイルの両方)、データセットオプション ASYNCINDEX=NO がデフォルトなので、並列作成されることはありません。

SPD Engine データセットが暗号化される場合は、データコンポーネントファイルのみ暗号化されます。メタデータコンポーネントファイルおよび両方のインデックスコンポーネントファイルは暗号化されません。

### PROC APPEND を使用したデフォルト Base SAS Engine データセットの変換

新しい SPD Engine データセットに対してデータセットオプションを指定する必要がある場合は、APPEND プロシジャを使用します。

次の例では、PROC APPEND を使用して、デフォルト Base SAS Engin データセットから SPD Engine データセットを作成します。ASYNCINDEX=YES データセットオプションでは、インデックスの並列作成が指定されます。PARTSIZE=オプションでは、100MB のパーティションの作成が指定されます。

```
libname spdelib spde 'new_data';
libname somelib 'old_data';
proc append base=spdelib.cars (asyncindex=yes partsize=100)
  data=somelib.cars;
run;
```

## 新しい SPD Engine データセットの作成とロード

新しい SPD Engine データセットを作成するには、DATA ステップ、任意の PROC ステートメント [1] と OUT=オプション、または PROC SQL と CREATE TABLE=オプションを使用します。

次の例では、DATA ステップを使用して、report_area ディレクトリに新しい SPD Engine データセット CARDATA.OLD_AUTOS を作成します。

```
libname cardata spde '/report_area';

data cardata.old_autos(compress=no encrypt=yes pw=secret);
  input year $4. @6 manufacturer $12. @18 model $12. @31 body_style $5. @37
  engine_liters @42 transmission_type $1. @45 exterior_color
  $10. @55 mileage @62 condition;

datalines;

1966 Ford       Mustang     conv  3.5  M  white     143000 2
1967 Chevrolet  Corvair      sedan 2.2  M  burgundy   70000 3
1975 Volkswagen  Beetle      2door 1.8  M  yellow     80000 4
```

[1] PROC MIGRATE 以外

```
1987 BMW       325is       2door 2.5  A  black    110000 3
1962 Nash      Metropolitan conv  1.3  M  red      125000 3
;
run;
```

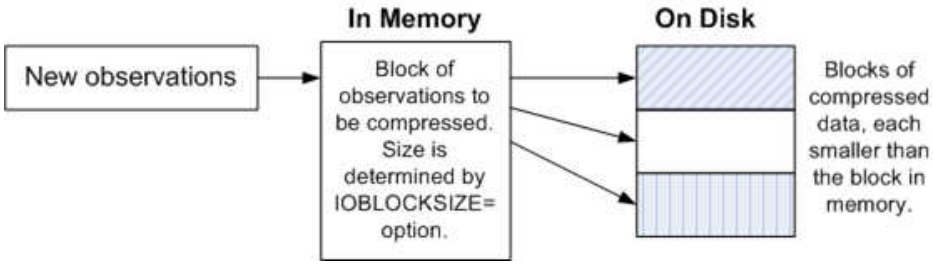
*注:* SPD Engine では、暗号化と圧縮は相互に排他的です。ENCRYPT=オプションを使用できるのは、圧縮されていない SPD Engine データファイルを作成する場合のみです。SPD Engine データセットを作成する際に暗号化と圧縮を両方とも行うことはできません。

## SPD Engine データセットの圧縮

COMPRESS=YES|BINARY|CHAR を指定した場合、SPD Engine では、データコンポーネントファイルが、作成時にブロックで圧縮されます。SPD Engine では、ユーザー指定の圧縮はサポートされません。さらに、圧縮と暗号化の両方が行われたデフォルト Base SAS Engine データセットをコピーする場合、暗号化は保持されますが、圧縮は解除されます。

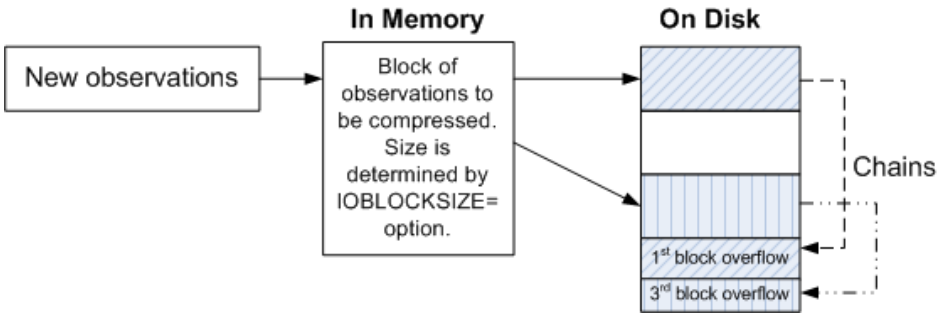
いったん圧縮データセットを作成すると、ブロックサイズは変更できません。圧縮ブロックは連続的に格納され、ブロック間に空き領域はありません。次の図は、ブロックのディスクへの格納方法を示しています。

**図 2.1** ディスク上の圧縮ブロック




圧縮後のデータセットへの更新で、ブロック内の利用可能容量よりも多くの領域が必要になった場合、SPD Engine ではオーバーフローを保持するために新しいブロックフラグメントが作成されます。さらに更新をして再度オーバーフローが発生した場合、新しいブロックフラグメントが作成され、チェーンが形成されます。次の図は、更新によるディスク上でのブロックチェーンの作成を示しています。

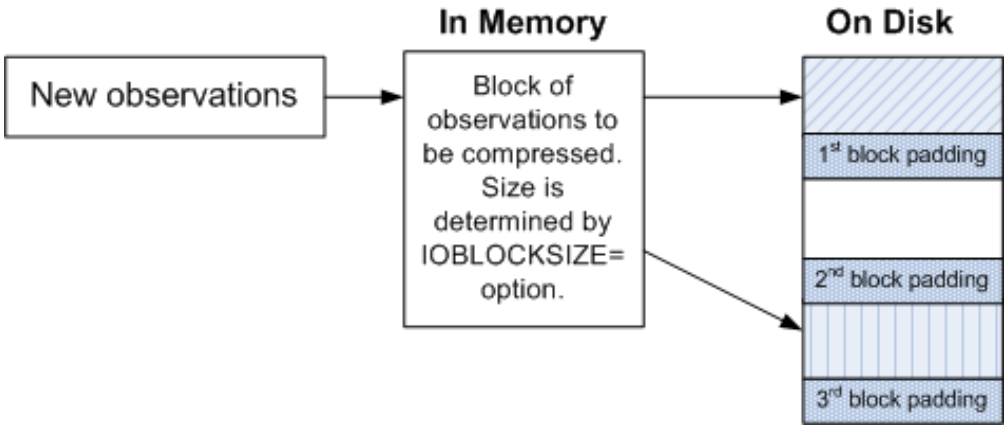
**図 2.2** オーバーフローのある圧縮ブロック



チェーンが長くなりすぎると、パフォーマンスに影響します。チェーンを削除してブロックのサイズ変更を行うには、データセットを新しいデータセットにコピーする必要があります。 IOBLOCKSIZE= (73 ページ) を、出力データセットに適したブロックサイズに設定します。

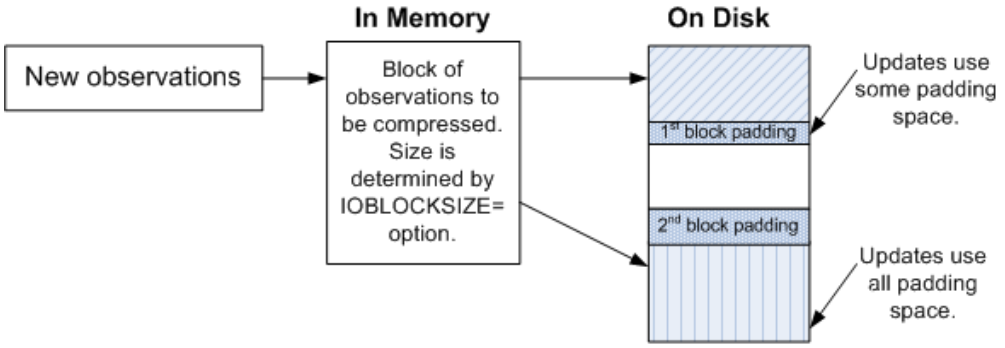
データセットの頻繁な更新が予想される場合、 PADCOMPRESS= (77 ページ)の使用をお勧めします。 SPD Engine では、新しいブロックフラグメントが作成されるかわりに、各ブロックに対する埋め込み領域が作成されます。次の図は、更新に対する各ブロックの領域の埋め込みを示しています。

**図 2.3** 圧縮された埋め込みブロック




圧縮後のデータセットへの更新で、ブロック内の利用可能容量よりも多くの領域が必要になった場合、SPD Engine は、各ブロックで埋め込み領域を使用します。新しいブロックフラグメントは作成されません。次の図は、更新による埋め込み領域の減少を示しています。

**図 2.4** 更新のある圧縮された埋め込みブロック




CONTENTS プロシジャでは、圧縮についての情報が印刷されます。次の例では、CONTENTS プロシジャ出力の圧縮情報フィールドについて説明します。

**アウトプット 2.1** CONTENTS プロシジャの圧縮情報出力

| - Compressed Info | - |
|---|---|
| Number of compressed blocks | 202 |
| Raw data blocksize | 32736 |
| Number of blocks with overflow | 5 |
| Max overflow chain length | 3 |
| Block number for max chain | 80 |
| Min overflow area | 87 |
| Max overflow area | 181 |

圧縮ブロック数
: データの格納に必要な圧縮ブロック数。

生データブロックサイズ
: IOBLOCKSIZE=データセットオプションに指定したサイズから計算されたバイト単位の圧縮ブロックサイズ。

オーバーフローのあるブロック数
: より多くの領域を必要とする圧縮ブロック数。データが更新されて、圧縮された新ブロックが圧縮された旧ブロックよりも大きくなった場合、オーバーフローブロックフラグメントが作成されます。

オーバーフローチェーンの最大長
: 単一ブロックのオーバーフロー最大数。たとえば、圧縮ブロックが更新されて大きくなり、再度更新されてより大きなサイズになった場合、オーバーフローチェーンの最大長は 2 になります。

最大チェーンのブロック数
: 最大数のオーバーフローブロックを含むブロック数。

最小オーバーフロー領域
: オーバーフローに必要なディスク領域最小容量。

最大オーバーフロー領域
: オーバーフローに必要なディスク領域最大容量。

# SPD Engine データセットの暗号化

## SPD Engine 暗号化の概要

*暗号化*とは、理解可能なデータ(プレーンテキスト)を理解不能な形式(暗号テキスト)へ数学的処理方法で変換することです。暗号テキストは、その暗号テキストの複合化(ロック解除)に必要となる適切なパスワードまたは ENCRYPTKEY が適合する場合に、翻訳されてプレーンテキストに戻されます。

暗号化は、ディスク上の情報と輸送中の情報の保護に役立ちます。

- *Over-the-wire* 暗号化は、輸送中の SAS データを保護します。

- *On-disk* 暗号化では、静止中のデータを保護します。

静止中の SPD Engine データセットの暗号化に SAS が使用するアルゴリズムは 2 種類あります。

SAS Proprietary
: Base SAS ソフトウェア内で提供されます。このアルゴリズムでは中程度レベルのセキュリティが提供されます。この暗号化を起動するには ENCRYPT=YES データセットオプションを使用します。

AES (Advanced Encryption Standard)
: 256 ビットキーを使用して 128 ビットのブロックとしてデータを暗号化する *ブロック暗号化*手法です。デフォルト Base SAS ソフトウェアに含まれている SAS/SECURE ソフトウェアを使用します。この暗号化を起動するには ENCRYPT=AES データセットオプションを使用します。

**表 2.2** SPD Engine 暗号化機能

| 機能 | ENCRYPT=YES | ENCRYPT=AES |
|---|---|---|
| ライセンスが必要 | いいえ | いいえ |
| 暗号化レベル | 中程度 | 高度 |
| サポートされるアルゴリズム | Base SAS ソフトウェア内 | AES |
| インストールが必要 | いいえ(Base SAS の一部) | いいえ(Base SAS に含まれている SAS/SECURE の一部) |
| サポートされる動作環境 | UNIX<br>Windows<br>z/OS | UNIX<br>Windows<br>z/OS |
| SAS のバージョンサポート | 8 以降 | 9.4 以降 |

## SAS Proprietary アルゴリズム

SAS Proprietary は、32 ビットの固定エンコーディングを使用するものであり、予期せぬ情報の開示を防ぐ場合にのみ適切なアルゴリズムとなります。SAS Proprietary のライセンスは Base SAS ソフトウェアに含まれているため、すべての配置で SASProprietary を利用できます。

## AES アルゴリズム

AES アルゴリズムは、256 ビットキーを使用して 128 ビットのブロックとしてデータを暗号化するブロック暗号化手法です。このアルゴリズムは、Base SAS に含まれている SAS/SECURE ソフトウェアを使用します。SAS/SECURE の詳細については、*Encryption in SAS* を参照してください。

*注:* AES 暗号化は、OpenVMS on 64–bit Itanium ではサポートされません。

SPD Engine データセットの拡張された暗号化機能を提供する AES 暗号化は、SAS9.4 以降で使用可能です。AES アルゴリズムは SAS/SECURE を使用して、より強力な暗号を作り出します。暗号化された SPD Engine データセットが必要な場合には、SPD Engine データセットの作成時に、ENCRYPT=AES を指定した ENCRYPTKEY=データセットオプションを使用する必要があります。

*注:* AES で暗号化された SPD Engine データセットでは ENCRYPTKEY=の値は変更できません。変更するにはデータセットの再作成が必要です。

次の規則が SPD Engine データセットの AES 暗号化に適用されます。

- AES 暗号化でデータセットを作成する場合は、ENCRYPTKEY=データセットオプションを使用する必要があります。
- AES で暗号化された SPD Engine データセットをコピーするには、出力エンジンで AES 暗号化がサポートされている必要があります。サポートされていない場合、データセットはコピーされません。
- SAS 9.4 より前のリリースでは、AES で暗号化された SPD Engine データセットは使用できません。
- SPD Engine データセットが AES 暗号化される場合、関連するインデックスファイルもすべて AES で暗号化されます。メタデータファイルは AES 暗号化されません。

詳細については、“ENCRYPT=データセットオプション” (62 ページ) および “ENCRYPTKEY=データセットオプション” (65 ページ) を参照してください。

## SPD Engine コンポーネントファイルの命名規則

SPD Engine データセットを作成する場合、多くのコンポーネントファイルも作成できます。SPD Engine コンポーネントファイルは次の命名規則で格納されます。

filename.mdf.0.p#.v#.spds9
filename.dpf.fuid.p#.v#.spds9
filename.idxsuffix.fuid.p#.v#.spds9
filename.hbxsuffix.fuid.p#.v#.spds9

*filename*
有効な SAS ファイル名。

mdf
メタデータコンポーネントファイルを識別します。

dpf
分割データコンポーネントファイルを識別します。

*p#*
パーティション番号です。

*v#*
バージョン番号です。[1]

*fuid*
一意のファイル ID。プライマリ(メタデータ)パスに相当する 16 進数。

[1] バージョン番号は、データセットが更新されるとき、すなわち、データセットが UPDATE モードで開くときのみ増加します。PROC SORT のようなデータセットを置換する操作では、バージョン番号が増すかわりに、1 にリセットされます。

idx*suffix*
インデックスのセグメント化ビューを識別します。このとき *suffix* はインデックス名です。

hbx*suffix*
インデックスのグローバルビューを識別します。このとき *suffix* はインデックス名です。

spds9
SAS 9 SPD Engine コンポーネントファイルを示します。

表 2.2 は、この LIBNAME ステートメントおよび DATA ステップの使用時に作成されるデータセットコンポーネントファイルを示しています。

```
libname sample spde '/DATA01/SAS-library'
  datapath=('/DATA01/mydir' '/DATA02/mydir')
  indexpath=('/IDX1/mydir');
data sample.mine(index=(ssn));
  do i=1 to 100000;
  ssn=ranuni(0);
  end;
run;
```

**表 2.3** データセットコンポーネントファイル

| | |
|---|---|
| mine.mdf.0.0.0.spds9 | メタデータコンポーネントファイル |
| mine.dpf.000032a6.0.1.spds9 | データファイルパーティション#1 |
| mine.dpf.000032a6.1.1.spds9 | データファイルパーティション#2 |
| mine.dpf.000032a6.*n*-1.1.spds9 | データファイルパーティション#*n* |
| mine.dpf.000032a6.*n*.1.spds9 | データファイルパーティション#*n*+1 |
| mine.hbxssn.000032a6.0.1.spds9 | 変数 SSN のグローバルインデックスデータセット |
| mine.idxssn.000032a6.0.1.spds9 | 変数 SSN のセグメント化インデックスデータセット |

# SPD Engine における効率的なインデックス付け

## インデックス付けの並列処理

インデックスによって、WHERE 式処理および BY 式処理のパフォーマンス向上が可能になります。SPD Engine では、並列処理ができるので、インデックスの迅速な作成および更新が可能になります。

SPD Engine のインデックスは特に、さまざまなサイズのデータセットおよびデータ分散に適しています。これらのインデックスには、インデックス付き変数値のセグメント化ビューとグローバルビューの両方が含まれます。この機能を使用すると、SPD Engine で、次のクエリの両方を最適にサポートできます。

- グローバルデータビューを必要とするクエリ(BY 式処理など)
- セグメント化ビューを必要とするクエリ(WHERE 式の並列処理など)

## 並列処理でのインデックス作成

SPD Engine データのインデックスは、非同期で並列作成できます。非同期並列インデックスの作成を可能にするには、“ASYNCINDEX=データセットオプション” (52 ページ)を使用します。

複数のインデックスを有するデータセットの作成時には、このオプションを、DATA ステップの INDEX=オプションおよび PROC DATASETS MODIFY ステートメントと一緒に使用します。どちらの方法でも、1 回のデータセットスキャンによってすべての宣言インデックスを事前設定することができます。

次の例は、DATA ステップを使用して並列作成されたインデックスを示しています。変数 X に単一インデックス、変数 A および B に複合インデックスが作成されます。

```
data foo.mine(index=(x y=(a b)) asyncindex=yes);
  x=1;
  a="Doe";
  b=20;
run;
```

複数のインデックスを並列作成するには、すべてのキー並べ替えを同時に行うのに十分なユーティリティディスク領域を割り当てる必要があります。十分なメモリ領域も割り当ててください。ディスク領域を割り当てるにはSPDEUTILLOC=システムオプション (99 ページ)を使用し、追加メモリを割り当てるには構成ファイル内か起動時に SPDEINDEXSORTSIZE システムオプション (96 ページ)を使用します。

DATASETS プロシジャには柔軟性があるので、複数の MODIFY グループを使用したバッチ並列インデックス作成が可能です。一度にすべてのインデックスを作成すると大量の領域が必要になりますが、そのかわりに、次の例に示すようにグループ単位でインデックスを作成できます。

```
proc datasets lib=main;
  modify patients(asyncindex=yes);
    index create number;
    index create class;
  run;
```

```
  modify patients(asyncindex=yes)'
    index create lastname firstname;
  run;
  modify patients(asyncindex=yes);
    index create fullname=(lastname firstname);
    index create class_sex=(class sex);
  run;
quit;
```

インデックス Number およびインデックス Class、インデックス LastName およびインデックス FirstName、ならびにインデックス FullName および Class_Sex がそれぞれ並列作成されます。

### 並列処理でのインデックス更新

SPD Engine では、データセット追加操作時の並列インデックス更新もサポートされています。複数のスレッドによって、データストアおよびインデックスファイルの更新が可能になります。SPD Engine では、データセットの追加操作または挿入操作が、並列実行の可能な一連のステップに分解されます。並列処理の到達レベルは、データセット内のインデックス数によって決まります。並列インデックス作成と同様に、この操作では、インデックス追加処理の一部であるキー並べ替えのためにメモリおよびディスク領域が使用されます。メモリを割り当てるにはシステムオプション SPDEINDEXSORTSIZE=、ディスク領域を割り当てるには SPDEUTILLOC=を使用します。

*注:* ASYNCINDEX オプションは、並列インデック更新に対しては無効です。

## SPD Engine ファイルのバックアップ

SPD Engine データセットをバックアップする場合は、次の要件に注意してください。

- データセットを構成するファイルはすべて、異なるディスクやファイルシステムに存在する場合でも、まとめて同時にバックアップするようにしてください。
- 更新中のファイルがある場合、データセットのバックアップは行わないでください。
- バックアップ後には毎回テストを実行し、バックアップが成功したか確認します。

## HDFS での SPD Engine データの格納

SPD Engine では、HDFS でデータの読み取り、書き込み、および更新を行えます。HDFS で SPD Engine データを格納することによって、ビッグデータを格納するための低コストの代替方法が提供されます。標準 SAS アプリケーションで SPD Engine を使用すると、分析用データの取得、管理機能の実行、およびデータの更新を行えます。

3 章

# SPD Engine LIBNAME ステートメントオプション



## SPD Engine LIBNAME ステートメントについて

このセクションには、SPD Engine の LIBNAME ステートメントに有効なすべての LIBNAME オプションのための参考情報が含まれています。

これらの LIBNAME オプションのうちのいくつかはデータセットオプションでもあります。デフォルト Base SAS Engine でのように、両方のオプションが設定された場合、対応する LIBNAME オプションよりデータセットオプションが優先します。

## 構文

LIBNAME *libref* SPDE '*full-primary-path*' <*option(s)*> ;

*libref*
SAS 命名規則に則った最大 8 文字長の名前。

*'full-primary-path'*
SPD Engine ライブラリ用プライマリパスの完全パス名。その名前が動作環境で認識されることが必要です。一重引用符または二重引用符で名前を囲みます。DATAPATH=オプションと INDEXPATH=オプションが指定されない場合は、インデックスとデータコンポーネントが同じ場所に格納されます。プライマリパスはライブラリごとに一意であることが必要です。同じプライマリパス名を参照しているが異なる Libdef は、同じライブラリであると解釈され、データが失われる場合があります。

*option(s)*
1 つ以上の SPD Engine LIBNAME ステートメントオプション。

*動作環境情報*: 有効なライブラリの指定と構文は、動作環境に固有です。詳細については、お使いの動作環境向けの SAS ドキュメントを参照してください。

## SPD Engine LIBNAME ステートメントオプションのリスト

ACCESS=READONLY
データセットの読み取りはできるが、更新や作成はできないことを指定します。

BYSORT=
BY ステートメントの検出時に SPD Engine で自動並べ替えを実行することを指定します。BYSORT=はデータセットオプションでもあります。

COMPRESS=
SPD Engine データセットの作成時にディスク上で圧縮するように指定します。COMPRESS=はデータセットオプションでもあります。

DATAPATH=
SPD Engine データセットのためのデータパーティション(.dpf)を格納するパスのリストを指定します。

ENDOBS=
ユーザー定義のオブザベーション処理範囲における最後のオブザベーション番号を指定します。ENDOBS=はデータセットオプションでもあります。

IDXBY=
SPD Engine での BY ステートメント処理時にインデックスを使用するかどうかを指定します。IDXBY=はデータセットオプションでもあります。

INDEXPATH=
SPD Engine データセットに関連付けられた 2 種類のインデックスコンポーネントファイル(.hbx および.idx)を格納するパスまたはパスのリストを指定します。

IOBLOCKSIZE=
I/O 操作で使用するオブザベーションブロックのサイズをバイト単位で指定します。IOBLOCKSIZE=はデータセットオプションでもあります。

METAPATH=
SPD Engine データセットのためのオーバーフローメタデータ(.mdf)コンポーネントファイルを格納するパスのリストを指定します。

PARTSIZE=
データコンポーネントパーティションがとられる最大サイズを指定します。値は SPD Engine データセットの作成時に指定されます。このサイズは固定

サイズです。この指定はデータコンポーネントファイルにのみ適用されます。PARTSIZE=はデータセットオプションでもあります。

STARTOBS=
ユーザー定義のオブザベーション処理範囲における最初のオブザベーション番号を指定します。STARTOBS=はデータセットオプションでもあります。

TEMP=
プライマリパスの一時サブディレクトリにライブラリを格納するよう指定します。

# ディクショナリ

## ACCESS= LIBNAME ステートメントオプション

データソースのアクセスレベルを決定します。

**デフォルト:** none

**エンジン:** SPD Engine のみ

### 構文

**ACCESS=**READONLY

### 必須引数

**READONLY**
データセットの読み取りはできるが、更新や作成はできないことを指定します。

### 詳細

このオプションを使用して、データソースへの書き込みを防ぎます。このオプションを省略すると、必要なデータソース権限がある場合には、データセットの読み込み、更新、および作成ができます。

## BYSORT= LIBNAME ステートメントオプション

BY ステートメントの検出時に SPD Engine で自動並べ替えを実行することを指定します。

**デフォルト:** YES

**操作:** “BYNOEQUALS=データセットオプション” (53 ページ)

**エンジン:** SPD Engine のみ

### 構文

**BYSORT=**YES | NO

### 必須引数

**YES**
事前にデータを並べ替えるかわりに、BY ステートメントを検出したときに BY 変数に基づいて自動的にデータを並べ替えるように指定します。

**NO**
データを BY 変数に基づいて並べ替えないように指定します。NO を指定する場合は、BY ステートメントの前にすでにデータが並べ替えられている必要があります。インデックスは使用されません。

## 詳細

デフォルト Base SAS Engine を使用した DATA または PROC ステップ処理では、インデックスがない場合やオブザベーションが順序どおりではない場合、BY ステートメントが発行される前にデータセットを並べ替える必要があります。それに対して、SPD Engine では、オブザベーションが順序どおりではない場合、デフォルトで、アプリケーションに返されるデータが並べ替えられます。並べ替えたデータセットが新規作成される PROC SORT とは異なり、SPD Engine の自動並べ替えでは、永久データセットは変更されず、データセットは新規作成されません。ただし、ユーティリティファイル領域は使用されます。詳細については、“SPDEUTILLOC=システムオプション” (99 ページ)を参照してください。

デフォルトは BYSORT=YES です。BYSORT=YES 引数によって自動並べ替えが有効になり、オブザベーションは BY グループ順序で出力されます。データセットオプション BYNOEQUALS=YES の場合、グループ内のオブザベーションは、データセットでの順序とは異なる順序で出力される可能性があります。データセット順序を保持するには BYNOEQUALS=NO を設定します。

BYSORT=NO 引数の場合は、BY ステートメントの前にすでにデータが並べ替えられている必要があります。並べ替えは、前の PROC SORT か、または BY 変数順序でのデータセットの作成によるものです。データセットが並べ替えられていない場合、エラーが発生します。BYSORT=NO の場合、グループ化されたデータはデータセット順序でアプリケーションに配信されます。インデックスは、BY 変数順序のオブザベーションの取得には使用されません。BYSORT=NO の場合、データセットオプション BYNOEQUALS=は無効になります。

LIBNAME ステートメントで BYSORT=オプションを指定する場合、PROC ステップまたは DATA ステップでの BYSORT=の指定によって上書きされる可能性があります。入力モードで開くために、LIBNAME ステートメントの BYSORT=NO を上書きするには、DATA ステップまたは PROC ステップで BYSORT=YES を設定します。重要な点は、BYSORT=NO によって、エンジンがデータの並べ替え処理をしないように指示されることです。

MSGLEVEL=I SAS システムオプションを設定した場合、BYSORT=YES および IDXWHERE=データセットオプションを使用すると、次のメッセージが SAS ログに書き込まれます。

- IDXWHERE=YES で BY 変数にインデックスがある場合、テーブルの行の並べ替えにそのインデックスが使用されます。次のメッセージが SAS ログに書き込まれます。

  Note: BY ordering was produced by using an index for table *tablename*.

- IDXWHERE=NO または IDXWHERE=YES で、BY 変数にインデックスがない場合、SPD Engine で自動並べ替えが実行され、テーブルの行が並べ替えられます。次のメッセージが SAS ログに書き込まれます。

  Note: BY ordering was produced by performing an automatic sort on table *tablename*.

## 比較

BYSORT=データセットオプションで、BYSORT= LIBNAME ステートメントオプションが上書きされます。

## 例

### 例 1: BYSORT=YES を使用するデフォルトのグループフォーマット

```
libname growth spde 'SAS-library';
data growth.teens;
  input Name $ Sex $ Age Height Weight;
datalines;
Alfred M 14 69.0 112.5
Carol F 14 62.8 102.5
James M 13 57.3 83.0
Janet F 15 62.5 112.5
Judy F 14 64.3 90.0
Philip M 16 72.0 150.0
William M 15 66.5 112.0
;
proc print data=growth.teens; by sex;
run;
```

BYSORT=YES がデフォルトのため、PROC SORT を使用してデータを並べ替えていなくても、エラーは発生しません。出力は次のとおりです。

**アウトプット 3.1** BYSORT=YES を使用するデフォルトのグループフォーマット

The SAS System

Sex=F

| Obs | Name | Age | Height | Weight |
|---|---|---|---|---|
| 2 | Carol | 14 | 62.8 | 102.5 |
| 4 | Janet | 15 | 62.5 | 112.5 |
| 5 | Judy | 14 | 64.3 | 90.0 |

Sex=M

| Obs | Name | Age | Height | Weight |
|---|---|---|---|---|
| 1 | Alfred | 14 | 69.0 | 112.5 |
| 3 | James | 13 | 57.3 | 83.0 |
| 6 | Philip | 16 | 72.0 | 150.0 |
| 7 | William | 15 | 66.5 | 112.0 |

### 例 2: LIBNAME ステートメントでの BYSORT=NO の使用

次の例では、DATA ステップまたは PROC ステップで BYSORT=YES が指定されず、LIBNAME ステートメントでの BYSORT=NO 指定が上書きされなかったので、SAS はエラーを返します。自動並べ替えが抑制される場合(BYSORT=NO)、(たとえば、PROC SORT を使用する)BY ステートメントの前にデータが BY 変数で並び替えられてる必要があります。

```
libname growth spde 'SAS-library' bysort=no;
proc print data=growth.teens;
by sex;
run;
```

```
ERROR: Data set GROWTH.TEENS is not sorted in ascending sequence.The current by-group has Sex = M and the next by-group has Sex = F. NOTE: The SAS System stopped processing this step because of errors.
```

## COMPRESS= LIBNAME ステートメントオプション

SPD Engine データセットの作成時にディスク上で圧縮するように指定します。

**デフォルト:** NO

**制限事項:** ENCRYPT=YES または ENCRYPT=AES と一緒には使用不可

**操作:** “IOBLOCKSIZE= LIBNAME ステートメントオプション” (39 ページ)
“PADCOMPRESS=データセットオプション” (77 ページ)

**エンジン:** SPD Engine のみ

### 構文

**COMPRESS=**NO | YES | CHAR | BINARY

### 必須引数

**NO**
データセットの圧縮は実行しません。

**YES | CHAR**
RLE (ランレングスエンコーディング)を使用して SPD Engine データセット内のデータをブロック単位で圧縮するように指定します。RLE では、反復実行される同じ文字(空白を含む)を 2 バイトまたは 3 バイトの表現に削減することでデータが圧縮されます。

**BINARY**
RDC (ロスデータ圧縮)を使用して SPD Engine データセット内のデータをブロック単位で圧縮するように指定します。RDC では、RLE とスライディングウィンドウ圧縮を組み合わせて反復バイトパターンをより効果的に表現することでファイルが圧縮されます。

*注:* これは、中容量から大容量(数百バイト以上)のブロックのバイナリデータ(文字変数と数値変数)を圧縮する場合に大きな効果を発揮する方法です。

## 詳細

COMPRESS=YES|BINARY|CHAR を指定した場合、SPD Engine では、データコンポーネントファイルが、作成時にブロックで圧縮されます。圧縮ブロックのサイズを指定するには、データセットの作成時に“IOBLOCKSIZE=データセットオプション” (73 ページ)を使用します。新しく圧縮したブロックに埋め込みを追加するには、データセットの作成または更新時に“PADCOMPRESS=データセットオプション” (77 ページ)を指定します。詳細については、“SPD Engine データセットの圧縮” (19 ページ)を参照してください。

SPD Engine では、ユーザー指定の圧縮はサポートされません。圧縮と暗号化の両方が行われたデフォルト Base SAS Engine データセットを移行する場合、暗号化は保持されますが、圧縮は解除されます。

CONTENTS プロシジャでは、圧縮設定が識別されます。データセットが圧縮される場合、PROC CONTENTS で圧縮についての情報が出力されます。次の例では、CONTENTS プロシジャ出力の圧縮情報フィールドについて説明します。

通常、COMPRESS=CHAR を指定すると、単一バイトの反復が存在する場合に適切な圧縮結果がもたらされ、COMPRESS=BINARY を指定すると、バイト列の反復が存在する場合に適切な圧縮結果がもたらされます。また、単一バイトの反復よりも、バイト列の反復を見つける方が労力を要します。たとえば、“例 1: COMPRESS=CHAR” (61 ページ)および“例 2: COMPRESS=BINARY” (62 ページ)を参照してください。

**アウトプット 3.2** PROC CONTENTS 圧縮情報セクション

| - Compressed Info | - |
|---|---|
| Number of compressed blocks | 202 |
| Raw data blocksize | 32736 |
| Number of blocks with overflow | 5 |
| Max overflow chain length | 3 |
| Block number for max chain | 80 |
| Min overflow area | 87 |
| Max overflow area | 181 |

圧縮ブロック数
: データの格納に必要な圧縮ブロック数。

生データブロックサイズ
: IOBLOCKSIZE=データセットオプションに指定したサイズから計算されたバイト単位の圧縮ブロックサイズ。これは、ブロックサイズ単位で取得するオブザベーション長の最大倍数です。

オーバーフローのあるブロック数
: より多くの領域を必要とする圧縮ブロック数。データが更新されて、圧縮された新ブロックが圧縮された旧ブロックよりも大きくなった場合、オーバーフローブロックフラグメントが作成されます。

オーバーフローチェーンの最大長
単一ブロックのオーバーフロー最大数。たとえば、圧縮ブロックが更新されて大きくなり、再度更新されてより大きなサイズになった場合、オーバーフローチェーンの最大長は 2 になります。

最大チェーンのブロック数
最大数のオーバーフローブロックを含むブロック数。

最小オーバーフロー領域
オーバーフローに必要なディスク領域最小容量。

最大オーバーフロー領域
オーバーフローに必要なディスク領域最大容量。

通常、圧縮ファイルへのアクセスには、より多くの処理時間を必要とします。読み取る前にファイルの圧縮解除が必要です。また、更新の場合は、ディスクに書き込む際に、もう一度圧縮する必要があります。

### 比較

COMPRESS= LIBNAME ステートメントオプションで、COMPRESS=システムオプションが上書きされます。

COMPRESS=データセットオプションで、COMPRESS= LIBNAME ステートメントオプションが上書きされます。

## DATAPATH= LIBNAME ステートメントオプション

SPD Engine データセットのためのデータパーティション(.dpf)を格納するパスのリストを指定します。

**デフォルト:** LIBNAME ステートメントに指定されたプライマリパス

**操作:** “PARTSIZE= LIBNAME ステートメントオプション” (42 ページ)
“PARTSIZE=データセットオプション” (78 ページ)

**エンジン:** SPD Engine のみ

### 構文

**DATAPATH=**(*'path1' <'path2'>*...)

#### 必須引数

***'path'***
一重引用符あるいは二重引用符で囲んだ完全パス名をかっこ内に指定します。複数の引数はスペースで区切ります。

*注:* DATAPATH=オプションに指定したパス名はライブラリごとに一意であることが必要です。同じパス名を参照しているが異なる Libdef では、データが失われる場合があります。

*注:* データが zFS ファイルシステムにある場合は、パス指定は 1 つのみで必須です。zFS システムは複数の論理ボリュームに渡るパーティションを自動的に展開します。

### 詳細

SPD Engine は、データをすべて格納するために必要な数だけのパーティションを作成します。パーティションのサイズは PARTSIZE=オプションを使用して設

定され、DATAPATH=オプションを使用して指定されたパスにパーティションが周期的に作成されます。

*注:* ローカルでマウントされたドライブにデータを格納し、リモートコンピュータからデータにアクセスする場合は、LIBNAME の指定時にリモートパス名を使用します。たとえば、**/data01** および**/data02** が localA コンピュータにローカルでマウントされたドライブである場合は、LIBNAME ステートメントでパス名**/nfs/localA/data01** および**/nfs/localA/data02** を使用します。

### 例: 最初のパーティションの DATAPATH=

最初のパーティションのパスはランダムに選択され、その後は周期的な方法で続きます。

```
libname mylib spde '/metadisk/metadata'
  datapath=('/disk1/dataflow1' '/disk2/dataflow2' '/disk3/dataflow3');
```

たとえば、/disk2/dataflow2 が最初のパスとしてランダムに選択されると、最初のパーティションはそこに置かれます。2 番目のパーティションは/disk3/dataflow3 に置かれ、3 番目のパーティションは/disk1/dataflow1 というように、配置されます。

## ENDOBS= LIBNAME ステートメントオプション

ユーザー定義のオブザベーション処理範囲における最後のオブザベーション番号を指定します。

**デフォルト:** データセット内の最後のオブザベーション

**制限事項:** ENDOBS=は入力データセットにのみ使用

OBS=システムまたはデータセットオプション、あるいは FIRSTOBS=システムまたはデータセットオプションと一緒には使用不可

**操作:** “ENDOBS=データセットオプション” (67 ページ)
“STARTOBS= LIBNAME ステートメントオプション” (44 ページ)
“STARTOBS=データセットオプション” (80 ページ)

**エンジン:** SPD Engine のみ

### 構文

**ENDOBS=*n***

#### 必須引数

***n***
終了オブザベーションの番号。

### 詳細

デフォルトでは、SPD Engine で、STARTOBS=オプションと ENDOBS=オプションを使用してオブザベーションの範囲を指定しない限り、データセット全体のすべてのオブザベーションが処理されます。STARTOBS=オプションを ENDOBS=オプションなしで使用した場合、ENDOBS=の暗黙値はデータセットの終わりになります。両方のオプションを一緒に使用する場合、ENDOBS=値を STARTOBS=値よりも大きくする必要があります。

デフォルト Base SAS Engine オプション FIRSTOBS=と違い、STARTOBS=および ENDOBS= SPD Engine オプションは LIBNAME ステートメントで使用できます。

*注:* OBS=システムオプションおよび OBS=データセットオプションは、STARTOBS=もしくは ENDOBS=データセットまたは LIBNAME オプションと一緒には使用できません。

(WHERE 処理での ENDOBS=データセットオプションの使用については、SPD Engine データセットオプション (47 ページ) を参照してください。)

## 比較

ENDOBS=データセットオプションで、ENDOBS= LIBNAME ステートメントオプションが上書きされます。

## 例: ENDOBS= LIBNAME ステートメントオプションの使用

次の例は、WHERE 句を実行する前に STARTOBS=オプションと ENDOBS=オプションがデータをサブセット化することを示します。例では、WHERE 式(PROC PRINT にある age >13)で適合した 4 つのオブザベーションを出力します。4 つのオブザベーションは、入力データセットから 5 つのオブザベーションが処理された結果です。

```
libname growth spde 'SAS-library' endobs=5;
data growth.teens;
  input Name $ Sex $ Age Height Weight;
datalines;
Alfred M 14 69.0 112.5
Carol F 14 62.8 102.5
James M 13 57.3 83.0
Janet F 15 62.5 112.5
Judy F 14 64.3 90.0
Philip M 16 72.0 150.0
William M 15 66.5 112.0
;
proc print data=growth.teens;
  where age >13;
run;
```

**アウトプット 3.3** ENDSOBS=

The SAS System

| Obs | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 1 | Alfred | M | 14 | 69.0 | 112.5 |
| 2 | Carol | F | 14 | 62.8 | 102.5 |
| 4 | Janet | F | 15 | 62.5 | 112.5 |
| 5 | Judy | F | 14 | 64.3 | 90.0 |

# IDXBY= LIBNAME ステートメントオプション

SPD Engine での BY ステートメント処理時にインデックスを使用するかどうかを指定します。

**デフォルト:** YES

**操作:** “BYSORT= LIBNAME ステートメントオプション” (29 ページ)
“BYSORT=データセットオプション” (57 ページ)

**エンジン:** SPD Engine のみ

## 構文

**IDXBY=**YES | NO

### 必須引数

**YES**
BY ステートメントのインデックス変数を処理するときにインデックスを使用します。

*注:* BY ステートメントに 2 つ以上の変数または DESCENDING オプションが指定されている場合は、IDXBY=YES でもインデックスは使用されません。

**NO**
BY ステートメントのインデックス変数を処理するときにインデックスを使用しません。

注 IDXBY=NO の場合、BY ステートメントの処理時に自動並べ替えが実行されます。

## 詳細

IDXBY= LIBNAME オプションを使用する場合は、BYSORT=YES オプションを使用し、BY 変数にインデックスを付けるようにしてください。

場合によっては、データを自動的に並べ替えると、SPD Engine のパフォーマンスが向上することがあります。自動並べ替えを使用するには、BYSORT=YES を設定し、IDXBY=NO を指定する必要があります。

SAS システムオプション MSGLEVEL=I を設定し、BY 処理情報が SAS ログに書き込まれるようにします。IDXBY=LIBNAME オプションおよび BYSORT=YES オプションを使用すると、SAS ログに次のメッセージが書き込まれます。

- IDXBY=YES で BY 変数にインデックスがある場合、テーブルの行の並べ替えにそのインデックスが使用されます。次のメッセージが SAS ログに書き込まれます。

  NOTE: BY ordering was produced by using an index for
  table *tablename*.

- IDXBY=NO を使用する場合、次のメッセージが SAS ログに書き込まれます。

  NOTE: BY ordering was produced by performing an automatic sort
  on table *tablename*.

### 比較

IDXBY=データセットオプションで、IDXBY= LIBNAME ステートメントオプションが上書きされます。

### 例

#### 例 1: IDXBY=NO LIBNAME オプションの使用

```
libname permdata spde 'SAS-library' idxby=no;
options msglevel=i;
  proc means data=permdata.customer;
    var sales;
  by state;
  run;
```

次のメッセージが SAS ログに書き込まれます。

```
NOTE: BY ordering was produced by performing an automatic sort on table PERMDATA.customer.NOTE:
There were 100 observations read from the data set PERMDATA.CUSTOMER.
```

#### 例 2: IDXBY=YES LIBNAME オプションの使用

次の例は IDXBY=YES を使用しています。

```
libname permdata spde 'SAS-library' idxby=yes;
options msglevel=i;
  proc means data=permdata.customer;
  var sales;
  by state;
run;
```

次のメッセージが SAS ログに書き込まれます。

```
NOTE: BY ordering was produced by using an index for table PERMDATA.customer.NOTE: There were
2981 observations read from the data set PERMDATA.CUSTOMER.
```

## INDEXPATH= LIBNAME ステートメントオプション

SPD Engine データセットに関連付けられた 2 種類のインデックスコンポーネントファイル(.hbx および.idx)を格納するパスまたはパスのリストを指定します。

**デフォルト:** LIBNAME ステートメントに指定されたプライマリパス

**エンジン:** SPD Engine のみ

### 構文

**INDEXPATH=**(*'path1' <'path2'...>*)

### 必須引数

***'path'***
一重引用符あるいは二重引用符で囲んだ完全パス名をかっこ内に指定します。複数の引数はスペースで区切ります。

*注:* INDEXPATH=オプションに指定したパス名はライブラリごとに一意であることが必要です。同じパス名を参照しているが異なる Libdef では、データが失われる場合があります。

## 詳細

メタデータコンポーネントファイルとは違って、インデックスコンポーネントファイルはプライマリパスに置く必要はありません。詳細については、“インデックスコンポーネントファイルの記憶域” (14 ページ)を参照してください。

INDEXPATH=オプションを使用するとインデックス I/O を別の物理パスやデバイスに移動させることができます。これによってパフォーマンスが向上します。詳細については、“I/O のパフォーマンスを向上させる機能” (9 ページ)を参照してください。

SPD Engine は、指定された場所に 2 つのインデックスコンポーネントファイルを作成します。INDEXPATH=オプションで複数のパスが指定されている場合、最初のパスはランダムに選択されます。複数のパスが指定された場合、インデックスコンポーネントファイルは最初のパスで作成され、最初のパスがいっぱいになると次のパスにオーバーフローします。

*注:* ローカルでマウントされたドライブにデータを格納し、リモートコンピュータからデータにアクセスする場合は、LIBNAME の指定時にリモートパス名を使用します。たとえば、**/data01** および**/data02** が localA コンピュータにローカルでマウントされたドライブである場合は、LIBNAME ステートメントでパス名**/nfs/localA/data01** および**/nfs/localA/data02** を使用します。

## 例: インデックスコンポーネントファイルの作成

次の例では、インデックスコンポーネントファイルを、パス/disk1/idxflow1、/disk2/idxflow2、および/disk3/idxflow3 にわたって作成します。

```
libname mylib spde '/metadisk/metadata'
            datapath= ('/disk1/dataflow1' '/disk2/dataflow2'
                    '/disk3/dataflow3')
            indexpath=('/disk1/idxflow1' '/disk2/idxflow2'
                    '/disk3/idxflow3' );
```

最初のインデックスコンポーネントファイルのパスはランダムに選択されます。SAS は、最初の場所にインデックスコンポーネントファイルを置き、その場所がいっぱいになるまで周期的にそこを使い続けます。たとえば、/disk2/idxflow2 がランダムに選択された場合は、そこに最初のインデックスコンポーネントファイルが置かれます。その場所がいっぱいになったときインデックスコンポーネントファイルは、/disk3/idxflow3、その後/disk1/idxflow1 にオーバーフローします。

# IOBLOCKSIZE= LIBNAME ステートメントオプション

I/O 操作で使用するオブザベーションブロックのサイズをバイト単位で指定します。

**デフォルト:** 1,048,576 バイト(1 MB)

**範囲:** 最小ブロックサイズは 32,768 バイトです。最大ブロックサイズは SPD Engine データパーティションファイルの半分のサイズです。

**エンジン:** SPD Engine のみ

**ヒント:** データセットの読み取り時に、ブロックサイズがパフォーマンスに大きな影響を及ぼすことがあります。データの大部分を取得する場合、ブロックサイズが大きい方がパフォーマンスが向上します。ただし、WHERE 処理などで、データのサブセットを取得する場合は、ブロックサイズが小さいほうがパフォーマンスが上がります。

## 構文

**IOBLOCKSIZE=***n*

### 必須引数

***n***
オブザベーションブロックのバイト単位でのサイズです。

## 詳細

I/O ブロックサイズによって、I/O 操作で物理的に一緒に転送されるデータ量が決定されます。ブロックサイズが大きいほど、I/O は少なくなります。SPD Engine では、メモリ内のブロックを使用して、データコンポーネントファイルでの読み書き対象のオブザベーションを収集します。IOBLOCKSIZE=オプションでは、ブロックのサイズが指定されます。(実際のサイズは、*n* バイトの指定サイズに収まるオブザベーション最大数に対応するように計算されます。したがって、実際のサイズはオブザベーションの長さの倍数になります)。

ブロックサイズは、圧縮データセット、非圧縮データセット、および暗号化データセットに対する I/O 操作に影響します。ただし、その影響はさまざまで、I/O 操作に依存します。

- 圧縮データセットの場合、ブロックサイズによって、一緒に圧縮されるオブザベーションの数が決定されます。これにより、読み取り操作と書き込み操作の両方について物理的に転送されるデータの量が決定されます。ブロックサイズはファイルの永久属性です。異なるブロックサイズを指定するには、データセットを新しいデータセットにコピーし、出力ファイルに対して新しいブロックサイズを指定する必要があります。圧縮データセットの場合、読み取り操作と書き込み操作のどちらでもブロックサイズが大きい方がパフォーマンスを向上させられます。
- 暗号化データセットの場合、ブロックサイズはファイルの永久属性です。
- 非圧縮データセットの場合、ブロックサイズによって、ディスクからメモリへのデータ読み取りに使用されるブロックのサイズが決定されます。ブロックサイズは、データのディスクへの書き込み時には、影響はありません。非圧縮データセットの場合、ブロックサイズはファイルの永久属性ではありません。つまり、実行する読み取り操作に基づいて異なるブロックサイズを指定できます。たとえば、ランダムに分散されたデータの読み取りや、データのサブセットの読み取りでは、ブロックサイズを小さくする必要があります。これは、小さいブロックにアクセスする方が、大きいブロックにアクセスするよりも速いためです。それとは対照的に、均等もしくは順番に分散されたデータ、またはデータセット全体のスキャンが必要なデータの読み取りは、ブロックサイズが大きいほうがより適切に動作します。

### 比較

IOBLOCKSIZE=データセットオプションで、IOBLOCKSIZE= LIBNAME ステートメントオプションが上書きされます。

### 例: IOBLOCKSIZE=の使用

```
/*IOBLOCKSIZE set to 64K */
libname employees spde 'SAS-library' ioblocksize=65536;

/*IOBLOCKSIZE set to 512M */
libname sales spde 'SAS-library' ioblocksize= 524288;
```

## METAPATH= LIBNAME ステートメントオプション

SPD Engine データセットのためのオーバーフローメタデータ(.mdf)コンポーネントファイルを格納するパスのリストを指定します。

**エンジン:** SPD Engine のみ

### 構文

**METAPATH=**(*'path1' <'path2'...>*)

#### 必須引数

**'*path*'**
一重引用符あるいは二重引用符で囲んだ完全パス名をかっこ内に指定します。複数の引数はスペースで区切ります。

### 詳細

ライブラリのすべてのデータセットに対するメタデータコンポーネントファイルは、プライマリパスでの指定と同じ場所に存在する必要があります。ライブラリに対して新しいデータセットが作成されるときに、プライマリパス内の領域がいっぱいになっている場合、SPD Engine はプライマリパスでメタデータコンポーネントファイルの作成を開始できません。エラーメッセージが表示され、作成操作が失敗します。詳細については、“メタデータコンポーネントファイルの記憶域” (14 ページ)を参照してください。

METAPATH=オプションは、メタデータコンポーネントファイルのオーバーフロー領域専用の領域を指定します。各データセットのメタデータコンポーネントファイルは、プライマリパスに存在する必要があります。プライマリパスがいっぱいになると、オーバーフローは METAPATH=場所に設定されます。

*注:* ローカルでマウントされたドライブにデータを格納し、リモートコンピュータからデータにアクセスする場合は、LIBNAME の指定時にリモートパス名を使用します。**/data01** および**/data02** が localA コンピュータにローカルでマウントされたドライブである場合は、LIBNAME ステートメントでパス名**/nfs/localA/data01** および**/nfs/localA/data02** を使用します。

# PARTSIZE= LIBNAME ステートメントオプション

データコンポーネントパーティションに指定可能な最大サイズ(MB、GB または TB 単位)を指定します。値は SPD Engine データセットの作成時に指定されます。このサイズは固定サイズです。この指定はデータコンポーネントファイルにのみ適用されます。

**デフォルト:** 128 MB

**操作:** “DATAPATH= LIBNAME ステートメントオプション” (34 ページ)
“MINPARTSIZE=システムオプション” (95 ページ)

**エンジン:** SPD Engine のみ

## 構文

**PARTSIZE=***n* | *n*M | *n*G | *n*T

### 必須引数

***n* | *n*M | *n*G | *n*T**
MB、GB または TB でのパーティションのサイズです。M、G または T なしで *n* を指定した場合のデフォルトは MB です。PARTSIZE=128 と PARTSIZE=128M は同じ意味です。最大値は、8,796,093,022,207MB です。

制限事項 この制限は、オペレーティングシステム z/OS、Linux SLES 9 x86 および Windows ファミリの 32 ビットホストにのみ適用されます。パーティションサイズが 2GB 以上のデータセットを作成した場合、SAS 9.2 以前のバージョンの SPD Engine では、そのデータセットを開くことはできません。次のエラーメッセージが SAS ログに書き込まれます。
**ERROR: Unable to open data file because its data representation differs from the SAS session data representation.**

## 詳細

SPD Engine データは、後で並列処理されるため複数パーティションに格納される必要があります。PARTSIZE=を指定することで、SPD Engine データファイルが指定サイズに分割されます。実際のパーティションサイズは、*n*MB、GB または TB の指定サイズに収まるオブザベーション最大数に対応するように計算されます。

SPD Engine データセットのデータ部分を固定サイズのファイルに分割(パーティション)することによって、ソフトウェアで一部の操作に高度なスケーラビリティが導入されます。SPD Engine では、スレッドを並列起動できます(たとえば、WHERE 評価に対して 1 パーティションにつき 1 スレッドまで)。また、別々のデータパーティションによって、スレッド間のファイルアクセス競合のオーバーヘッドなしでデータを処理できます。各パーティションが 1 ファイルであるため、小さなパーティションサイズの代償として、オブザベーションを格納するためにファイル(UNIX i ノードなど)の数の増加が必要になります。

PARTSIZE=を使用したスケーラビリティ制限は、DATAPATH=オプションで指定したファイルシステムを、どのように構成し、どのように複数のストライプボリュームに分散させるのかによって決まります。(個々のボリュームのストライピング構成を、ディスクアレイの複数のディスクコントローラまたは SCSI チャネルに分散させる必要があります。)構成の目的は、データ取得時の並列処理を最大限にすることです。ディスクストライピングの詳細については、Scalability and

Performance の“SPD Engine”の下に示されている“I/O Setup and Validation” (http://support.sas.com/rnd/scalability)を参照してください。

PARTSIZE=の指定は SPD Engine システムオプション MINPARTSIZE=によって制限されます。このシステムオプションは通常システム管理者によって設定、保守されます。

パーティションサイズによって、データセット全体のスキャンを必要とする並列操作の多くの作業単位が決定されます。ただし、パーティションの増加が必ずしも処理の迅速化を意味するとは限りません。この折り合いをつけるには、データセットの格納に必要な物理ファイル(パーティション)の増加数と、パーティションの増加によって並列処理が可能になる作業量とのバランスをとる必要があります。パーティションが多くなると、データセットを処理するために開くファイルも多くなりますが、各パーティションのオブザベーション数は少なくなります。一般的なルールでは、パーティションは、1 データパスにつき 10 以下、1CPU につき 3 から 4 までとなります。

新しい SPD Engine データセットの適切なパーティションサイズを決定するには、次の点に注意する必要があります。

- データに対して実行するアプリケーションの種類
- データ量はどの程度か
- アプリケーションで使用可能な CPU はいくつあるか
- パーティションの格納に使用可能なディスクはどれか
- そのディスクと CPU との関係

たとえば、各 CPU の制御するディスクが 1 つだけの場合、適切なパーティションサイズは、各ディスクにほぼ同じ量のデータが含まれるサイズです。各 CPU の制御するディスクが 2 つの場合、適切なパーティションサイズは、負荷のバランスがとれているサイズです。各 CPU で、ほぼ同じ量の作業が行われます。

*注:* データセットの PARTSIZE=値は、データセットの作成後は変更できません。PARTSIZE=を変更するには、データセットを再作成し、LIBNAME ステートメント内か、またはその新しい(出力)データセットに対して、異なる PARTSIZE=値を指定する必要があります。

## 比較

PARTSIZE=データセットオプションで、PARTSIZE= LIBNAME ステートメントオプションが上書きされます。

## 例: パーティションサイズの指定

LIBNAME ステートメントでパーティションサイズを指定する場合、そのライブラリに格納されるデータセットのほとんどに適切なサイズを選択する必要があります。たとえば、8 つのディスク構成を持っていると仮定します。最小のデータセットには 20 ギガバイトのデータがあり、最大のものには 50 ギガバイトのデータがあります。また、残りのデータセットにはそれぞれ 36 ギガバイトのデータがあります。1250M のパーティションサイズは、36 ギガバイトのデータセット(1 つのディスクあたり 4 つのパーティション)には最適です。20 ギガバイトのデータセットは、1 つのディスクあたり 2 つのパーティションを使用し、50 ギガバイトのデータセットは、1 つのディスク当たり 5 つのパーティションを使用します。

```
libname sales spde '/primdisk' partsize=1250M
datapath=('/disk01' '/disk02' '/disk03' '/disk04'
'/disk05' '/disk06' '/disk07' '/disk08');
```

# STARTOBS= LIBNAME ステートメントオプション

ユーザー定義のオブザベーション処理範囲における開始オブザベーション番号を指定します。

**デフォルト:** データセット内の最初のオブザベーション

**制限事項:** STARTOBS=は入力データセットにのみ使用
OBS=システムまたはデータセットオプション、あるいは FIRSTOBS=システムまたはデータセットオプションと一緒には使用不可

**操作:** “STARTOBS=データセットオプション” (80 ページ)
“ENDOBS= LIBNAME ステートメントオプション” (35 ページ)
“ENDOBS=データセットオプション” (67 ページ)

**エンジン:** SPD Engine のみ

## 構文

**STARTOBS=*n***

### 必須引数

***n***
開始オブザベーションの番号。

## 詳細

デフォルトでは、SPD Engine で、STARTOBS=オプションと ENDOBS=オプションを使用してオブザベーションの範囲を指定しない限り、データセット全体のすべてのオブザベーションが処理されます。ENDOBS=オプションを STARTOBS=オプションなしで使用した場合、STARTOBS=の暗黙値は 1 になります。両方のオプションを一緒に使用する場合、STARTOBS=値を ENDOBS=値よりも小さくする必要があります。

デフォルト Base SAS Engine オプション FIRSTOBS=と違い、STARTOBS=および ENDOBS= SPD Engine オプションは LIBNAME ステートメントで使用できます。

*注:* FIRSTOBS=デフォルト Base SAS Engine オプションは、SPD Engine ではサポートされていません。OBS=システムオプションおよび OBS=データセットオプションは、STARTOBS=もしくは ENDOBS=データセットまたは LIBNAME オプションと一緒には使用できません。

(WHERE 処理での STARTOBS=データセットオプションの使用については、SPD Engine データセットオプション (47 ページ) を参照してください。)

## 比較

STARTOBS=データセットオプションで、STARTOBS= LIBNAME ステートメントオプションが上書きされます。

## 例: WHERE 式の使用

次の例では、WHERE 式(PROC PRINT にある age >13)で適合した 5 つのオブザベーションを出力します。5 つのオブザベーションは、データセットにある 2 番

目のオブザベーションから開始して 6 つのオブザベーションが処理された結果です。

```
libname growth spde 'SAS-library' startobs=2;
data growth.teens;
  input Name $ Sex $ Age Height Weight;
datalines;
Alfred M 14 69.0 112.5
Carol F 14 62.8 102.5
James M 13 57.3 83.0
Janet F 15 62.5 112.5
Judy F 14 64.3 90.0
Philip M 16 72.0 150.0
William M 15 66.5 112.0
;
proc print data=growth.teens;
  where age >13;
run;
```

出力は次のとおりです。

**アウトプット 3.4** STARTOBS=

The SAS System

| Obs | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 2 | Carol | F | 14 | 62.8 | 102.5 |
| 4 | Janet | F | 15 | 62.5 | 112.5 |
| 5 | Judy | F | 14 | 64.3 | 90.0 |
| 6 | Philip | M | 16 | 72.0 | 150.0 |
| 7 | William | M | 15 | 66.5 | 112.0 |

## TEMP= LIBNAME ステートメントオプション

プライマリパスの一時サブディレクトリにライブラリを格納するよう指定します。

**デフォルト:** NO

**エンジン:** SPD Engine のみ

### 構文

**TEMP=**YES | NO

### 必須引数

**YES**
一時サブディレクトリを作成する指定です。

**NO**
一時サブディレクトリを作成しない指定です。

## 詳細

TEMP=オプションでは、LIBNAME ステートメントで指定されたプライマリディレクトリの一時サブディレクトリを作成します。そのサブディレクトリとすべてのコンポーネントファイルがセッションの終わりに削除されます。

SAS オプション USER=と一緒に TEMP=を使用すると、単一レベル名で参照できる一時データセットを格納できます。

*注:* SAS/CONNECT ソフトウェアで SIGNON ステートメントを使用する場合、INHERITLIB=オプションでは、TEMP=オプションで定義された SPD Engine ライブラリを参照できません。

## 例: 一時ライブラリの作成

次の例では 2 つの機能を説明します。

- TEMP= LIBNAME オプションを使用して一時ライブラリを作成します。
- USER=システムオプションの使用によって、SPD Engine テーブルのための単一レベルテーブル名が使用可能になります。

MyData の下にディレクトリが作成されます。MasterCopy データセットには、MyData のサブディレクトリに格納されているメタデータファイルがあります。MasterCopy のためのデータとインデックスは、DATAPATH=オプションおよび INDEXPATH=オプションで指定した場所に作成されます。

```
libname perm <masterdata>
libname mywork spde 'mydata'
  datapath=('/data01/mypath' '/data02/mypath' '/data03/mypath' '/data04/mypath')
  indexpath=('index/mypath') TEMP=YES;
option user=mywork;
data mastercopy (index=(lastname));
  set perm.customer;
  where region='W';
run;
```

4 章
# SPD Engine データセットオプション



## SPD Engine データセットオプションについて

SPD Engine のデータセットオプションの指定は、デフォルト Base SAS Engine または SAS/ACCESS Engine のデータセットオプションの指定と同じです。このセクションでは、SPD Engine でだけ使用されるデータセットオプションに関する詳細を提供します。SPD Engine に影響があるデフォルト Base SAS Engine データセットオプションもリストしています。

オプションの使用時には、データセットオプションの値で、対応する LIBNAME オプションの値が上書きされることに注意してください。

## 構文

(*option-1=value-1* <(*option-2=value-2*>...)

SAS データセット名の後に、データセットオプションをかっこで囲んで指定します。複数のデータセットオプションを指定するには、スペースで区切ります。

## SPD Engine データセットオプションのリスト

ALIGN=
変数アラインメントを指定します。

ASYNCINDEX=
SPD Engine データセットに複数のインデックスを作成する際、インデックスを並列作成するように指定します。

BYNOEQUALS=
BY 変数に同一の値を持っているデータセットオブザベーションのインデックス出力順序を指定します。

BYSORT=
BY ステートメントの検出時に SPD Engine で自動並べ替えを実行することを指定します。BYSORT=は LIBNAME ステートメントオプションでもあります。

COMPRESS=
SPD Engine データセットの作成時にディスク上で圧縮するように指定します。COMPRESS=は LIBNAME ステートメントオプションでもあります。

*注:* SPD Engine では、圧縮と暗号化は相互排他的です。

ENCRYPT=
出力 SPD Engine データセットを暗号化するかどうかを指定します。

*注:* SPD Engine では、圧縮と暗号化は相互排他的です。

ENCRYPTKEY=
AES 暗号化のキー値を指定します。

ENDOBS=
ユーザー定義のオブザベーション処理範囲における最後のオブザベーション番号を指定します。ENDOBS=は LIBNAME ステートメントオプションでもあります。

IDXBY=
SPD Engine での BY ステートメント処理時にインデックスを使用するかどうかを指定します。IDXBY=は LIBNAME ステートメントオプションでもあります。

IDXWHERE=
SPD Engine での WHERE 式処理時にインデックスを使用するかどうかを指定します。

IOBLOCKSIZE=
圧縮されるオブザベーションブロックのサイズをバイト単位で指定します。IOBLOCKSIZE=は LIBNAME ステートメントオプションでもあります。

LISTFILES=
CONTENTS プロシジャで、SPD Engine データセットのコンポーネントファイルすべての完全パス名をリストするかどうかを指定します。

PADCOMPRESS=
OUTPUT または UPDATE モードで開かれるデータセットの圧縮ブロックに追加するバイト数を指定します。

PARTSIZE=
データコンポーネントパーティションがとられる最大サイズを指定します。PARTSIZE=は LIBNAME ステートメントオプションでもあります。

STARTOBS=
ユーザー定義のオブザベーション処理範囲における最初のオブザベーション番号を指定します。STARTOBS=は LIBNAME ステートメントオプションでもあります。

SYNCADD=
一度に 1 つのオブザベーションを処理するのか、それとも一度にオブザベーションをブロックで処理するのかを指定します。

THREADNUM=
SPD Engine 処理に使用するスレッドの最大数を指定します。

UNIQUESAVE=
SYNCADD=NO の場合で、固有のインデックスを持つデータセットに追加あるいは挿入の際に、固有キー値ではないという理由で拒否された任意のオブザベーションを(別のファイルに)保管するために指定します。

WHERENOINDEX=
WHERE 式を評価するときに除外するインデックスのリストを指定します。

---

# SPD Engine とデフォルト Base SAS Engine で動作が異なる SAS データセットオプション

CNTLLEV=
値 MEM だけが使用できます。

COMPRESS=
ユーザー提供の値は使用できません。

**注意:**
**SPD Engine では、圧縮と暗号化は相互排他的です。** デフォルト Base SAS Engine データセットを SPD Engine データセットにコピーしようとするとき、そのデータセットが圧縮かつ暗号化されていた場合、圧縮は解除されます。SPD Engine データセットを作成する際に暗号化と圧縮を両方とも行うことはできません。

DLDMGACTION=
DLDMGACTION=NOINDEX はサポートしていませんが、ABORT、 FAIL、 PROMPT、REPAIR はサポートしています。

ENCRYPT=
データファイルを暗号化します。

**注意:**
**SPD Engine では、圧縮と暗号化は相互排他的です。**

## SPD Engine ではサポートされない SAS データセットオプション

- BUFNO=
- BUFSIZE=
- ENCODING=
- EXTENDOBSCOUNTER=
- FIRSTOBS=
- GENMAX=
- GENNUM=
- IDXNAME=
- OUTREP=
- POINTOBS=
- REUSE=
- TOBSNO=

## ディクショナリ

### ALIGN=データセットオプション

変数アラインメントを指定します。

**該当要素:** DATA ステップおよび PROC ステップ
**デフォルト:** YES
**制限事項:** SPD Server でのみ使用
**エンジン:** SPD Engine のみ

#### 構文

**ALIGN=**YES | NO

#### 必須引数

**YES**
変数アラインメントを有効化します。

**NO**
変数アラインメントを無効化し、SPD Engine データセットが SPD Server と互換性を持つようにします。

## 詳細

### 変数アラインメント

Base SAS は、オブザベーション内の変数を注意深く配列することにより、ディスク上のオブザベーションにおいて適切な数値データアラインメントを提供します。デフォルト Base Sas Engine と同様に、SPD Engine は、すべての数値がオブザベーションの先頭にまとめられることを保証します。また、Base SAS Engine は、必要に応じてオブザベーションの末尾に余分なパディングバイトを追加することにより、オブザベーション全体の長さが 8 バイトの偶数倍になることを保証します。SAS メモリシステムは、すべての割り当てられたメモリが 8 バイト境界で開始されること、およびすべての数値が 8 バイト長になることを保証します。数値をオブザベーションの先頭にまとめることにより、それらの数値が適切にアラインメントされ、同数値がメモリから直接使用可能となることが保証されます。

SPD Engine データセット内のオブザベーションは、ディスクからメモリに読み取られます。すべてのバイトが一度に読み取られるため、それらの相対的なアラインメントはメモリ内で維持されます。オブザベーションのデータへのアクセス時に最大のパフォーマンスを発揮するには、すべての数値データが適切にアラインメントされる必要があります。このため、値を新しい場所に移動する必要性が回避されます。SPD Engine の通常の動作は、ディスクへの書き出し時に、オブザベーション内のすべての数値が 8 バイト境界でアラインメントされることを保証します。これにより、SPD Engine はオブザベーションデータをディスクから取り出し、同データをメモリに格納できます。SPD Engine アプリケーションは、データを移動する必要なしに 、指定の格納場所からオブザベーションデータを直接使用します。

### SPD Server の使用

SPD Server はデータアラインメント機能をサポートしません。SPD Engine により作成され、 SPD Server データセットにより使用されるデータセットは、データアラインメントを無効にして作成する必要があります。ALIGN=NO オプションを指定すると、データセット内に格納されるデータはアラインメントされません。この場合、データをそれが読み取られた元々のメモリ上の場所から、アラインメント先となる異なるメモリ上の場所へと移動する必要があります。

ALIGN=NO データセットオプションは、SPD Engine によりデータアラインメントが行われないことを指定します。このオプションはまれにしか使用されませんが、SPD Server を使用する場合には必要となります。SPD Server は変数アラインメント機能をサポートしません。状況によっては、SPD Server は、アラインメントされた変数を含むデータセットに関する操作を拒否する場合もあります。

**注意:**

**データセットの宛先が SPD Server である場合を除き、ALIGN=NO オプションは使用しないでください。** 変数データがアラインメントされていない場合、Base SAS によって処理される際にパフォーマンスが大幅に低下します。

## 例

### 例 1: 変数アラインメントが有効であるデータセット

最初のデータセットは、アラインメントされた変数を含んでおり、デフォルトの動作を示します。オブザベーション長が変数長の合計と等しくならないことに注意してください。オブザベーション長は、8 の倍数に切り上げられています。

また、変数は、数値変数がオブザベーションの先頭に来るように並べ替えられています。

**ログ 4.1** 変数アラインメントが有効であるデータセット

```
data testdata.size; length text $10   width 8    chars 8; text='Zero';     width=1;   chars=4;  output;
text='Ten';      width=2;   chars=3;  output; text='Twenty';    width=2;   chars=6;  output; run; NOTE: The
data set TESTDATA.SIZE has 3 observations and 3 variables. proc sql; select obslen from
dictionary.tables where memname="SIZE"; Observation Length ---------- 32 select  varnum, name, npos,
length from dictionary.columns where memname="SIZE" order by npos; Column
Number                          Column    Column in Table  Column Name                Position    Length
---------------------------------------------------------------- 2  width                          0       8 3
chars                           8       8 1  text                          16       10 quit; /*
---------------------------------------------------------------- */ data testdata.size (align=no); length text $10   width
8    chars 8; text='Zero';     width=1;   chars=4;  output; text='Ten';      width=2;   chars=3;  output;
text='Twenty';    width=2;   chars=6;  output; run; NOTE: The data set TESTDATA.SIZE has 3 observations
and 3 variables.
```

### 例 2: 変数アラインメントが無効であるデータセット

2 番目のデータセットは、変数がアラインメントされていないことを除き、最初のデータセットと同じものです。オブザベーション長が変数長の合計と等しくなることに注意してください。オブザベーション長は、8 の倍数に切り上げられません。変数は、DATA ステップの LENGTH ステートメントに記述されている順番に並べられます。変数は、アラインメント用には並べ替えられていません。

**ログ 4.2** 変数アラインメントが無効であるデータセット

```
data testdata.size (align=no); length text $10   width 8    chars 8; text='Zero';     width=1;   chars=4;
output; text='Ten';      width=2;   chars=3;  output; text='Twenty';    width=2;   chars=6;  output; run;
NOTE: The data set TESTDATA.SIZE has 3 observations and 3 variables. proc sql; select obslen from
dictionary.tables where memname="SIZE"; Observation Length ---------- 26 select varnum, name, npos,
length from dictionary.columns where memname="SIZE" order by npos; Column
Number                          Column    Column in Table  Column Name                Position    Length
---------------------------------------------------------------- 1  text                           0       10 2
width                           10       8 3  chars                         18       8 quit; NOTE: The data set
TESTDATA.SIZE has 3 observations and 3 variables.
```

## ASYNCINDEX=データセットオプション

SPD Engine データセットに複数のインデックスを作成する際、インデックスを並列作成するように指定します。

**該当要素:** DATASETS プロシジャ、または INDEX データセットオプションと一緒に使用

**デフォルト:** NO

**エンジン:** SPD Engine のみ

### 構文

**ASYNCINDEX=**YES | NO

#### 必須引数

**YES**
インデックスを並列作成します(非同期)。

**NO**
一度に 1 つのインデックスを作成します(同期)。

## 詳細

SPD Engine では、1 回のデータセットスキャンで複数のインデックスを作成できます。SPD Engine では、作成されたインデックスごとに 1 つずつスレッドが起動され、そのスレッドが同時に処理されます。各インデックスのデータセットをスキャンするよりも、インデックスを並列作成する方がずっと迅速に処理できますが、このオプションのデフォルトは NO です。これは、並列インデックス作成では、並べ替えファイルを格納するための追加ユーティリティ領域、および追加メモリが必要になるためです。リソース不十分のためインデックス作成が失敗した場合は、次のどちらかまたは両方を実行します。

- SPDEUTILLOC=システムオプションを使用して、ユーティリティファイル領域のサイズを増やします。
- SAS システムオプションを MEMSIZE=0 に設定し、[1] SPDEINDEXSORTSIZE=システムオプションを使用して、インデックスの並べ替えに使用されるユーティリティ領域を増やします。

## 例: グループ単位でのインデックス作成

DATASETS プロシジャには柔軟性があるので、複数の MODIFY グループを使用したバッチ並列インデックス作成を行えます。一度にすべてのインデックスを作成すると大量の領域が必要になりますが、そのかわりに、次の例に示すようにグループ単位でインデックスを作成できます。インデックス PatientNo および PatientClass は、インデックス LastName および FirstName と同様に、一緒に作成されます。その他のインデックスは連続的に作成されます。

```
proc datasets lib=main;
  modify patients(asyncindex=yes);
    index create PatientNo PatientClass;
  run;
  modify patients(asyncindex=yes);
    index create LastName FirstName;
  run;
  modify patients(asyncindex=no);
    index create FullName=(LastName FirstName)
      ClassSex=(PatientClass PatientSex);
  run;
quit;
```

# BYNOEQUALS=データセットオプション

BY 変数に対する同一値があるデータセットオブザベーションの出力順序がデータセット順序になることを保証するかどうかを指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**使用要素:** BYSORT=YES データセットオプション

**デフォルト:** NO

**エンジン:** SPD Engine のみ

[1] z/OS の場合、REGION サイズを増やします。

## 構文

**BYNOEQUALS=**YES | NO

### 必須引数

**YES**
BY 変数に対する同一値があるデータセットオブザベーションの出力順序がデータセット順序になることを保証しません。

**NO**
BY 変数に対する同一値があるデータセットオブザベーションの出力順序がデータセット順序になることを保証します。

## 詳細

BY ステートメントに対して同一値があるオブザベーションのグループが出力される場合、その出力でのオブザベーションの順序はデータセット順序と同じになります。これは、デフォルトが BYNOEQUALS=NO であるためです。YES を指定すると、処理時間は短縮されますが、オブザベーションがデータセット順序で出力される保証はありません。

BYNOEQUALS=オプションは BYSORT=NO の場合は無効になるので、データセットまたは LIBNAME オプション BYSORT=は YES (デフォルト)にする必要があります。

次の表は、SPD Engine でどのような場合に出力の物理的な順序が保持されるかを示しています。

**表 4.1** SPD Engine での物理的な順序の保持

| 条件: | データセット順序保持の有無 |
|---|---|
| BY が存在する場合 | あり(デフォルトで BYNOEQUALS=NO および BYSORT=YES) |
| BY が存在し、BYNOEQUALS=YES の場合 | NO |
| BY が存在し、BYSORT=NO の場合 | あり(自動並べ替えが発生しないため) |
| BY も WHERE も存在しない場合 | YES |
| WHERE が存在する場合 | NO |

## 例

### 例 1: BYNOEQUALS=YES

次の例では、BYNOEQUALS=YES のため、キー変数に同一の BY 値があるオブザベーションが予測不能な順序で出力されます。

```
title 'With BYNOEQUALS=YES';
proc print data=labs.performance(bynoequals=yes) noobs;
```

```
  by score;
run;
```

出力は次のとおりです。

**アウトプット 4.1** BYNOEQUALS=YES

With BYNOEQUALS=YES

Score=High

| Hours | Section |
|---|---|
| 15 | C |
| 32 | C |
| 3 | A |
| 18 | C |
| 27 | D |
| 6 | D |
| 37 | D |
| 0 | D |
| 17 | A |
| 4 | A |

Score=Low

| Hours | Section |
|---|---|
| 22 | C |
| 4 | C |
| 22 | A |
| 9 | C |
| 22 | A |
| 21 | A |
| 21 | A |
| 4 | D |
| 21 | D |
| 1 | C |

## 例 2: BYNOEQUALS=NO

次の例は、BYNOEQUALS=NO の出力を示しています。

```
title 'With BYNOEQUALS=NO;
proc print data=labs.performance(bynoequals=no) noobs;
```

```
  by score;
run;
```

出力は次のとおりです。

**アウトプット 4.2** BYNOEQUALS=NO

**With BYNOEQUALS=NO**

**Score=High**

| Hours | Section |
|---|---|
| 0 | C |
| 0 | A |
| 0 | D |
| 0 | A |
| 0 | D |
| 0 | D |
| 1 | A |
| 1 | D |
| 1 | A |
| 1 | C |

**Score=Low**

| Hours | Section |
|---|---|
| 0 | A |
| 0 | C |
| 0 | A |
| 0 | C |
| 0 | A |
| 0 | A |
| 1 | A |
| 1 | C |
| 1 | A |
| 1 | C |

# BYSORT=データセットオプション

BY ステートメントの検出時に SPD Engine で自動並べ替えを実行することを指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**デフォルト:** YES

**操作:** “BYNOEQUALS=データセットオプション” (53 ページ)

**エンジン:** SPD Engine のみ

## 構文

**BYSORT=**YES | NO

### 必須引数

**YES**
事前にデータを並べ替えるかわりに、BY ステートメントを検出したときに BY 変数に基づいて自動的にデータを並べ替えるように指定します。

**NO**
データを BY 変数に基づいて並べ替えないように指定します。NO を指定する場合は、BY ステートメントの前にすでにデータが並べ替えられている必要があります。インデックスは使用されません。

## 詳細

デフォルト Base SAS Engine を使用した DATA または PROC ステップ処理では、インデックスがない場合やオブザベーションが順序どおりではない場合、BY ステートメントが発行される前にデータセットを並べ替える必要があります。それに対して、SPD Engine では、オブザベーションが順序どおりではない場合、デフォルトで、アプリケーションに返されるデータが並べ替えられます。並べ替えたデータセットが新規作成される PROC SORT とは異なり、SPD Engine の自動並べ替えでは、永久データセットは変更されず、データセットは新規作成されません。ただし、ユーティリティファイル領域は使用されます。詳細については、“SPDEUTILLOC=システムオプション” (99 ページ)を参照してください。

デフォルトは BYSORT=YES です。BYSORT=YES 引数によって自動並べ替えが有効になり、オブザベーションは BY グループ順序で出力されます。データセットオプション BYNOEQUALS=YES の場合、グループ内のオブザベーションは、データセットでの順序とは異なる順序で出力される可能性があります。データセット順序を保持するには BYNOEQUALS=NO を設定します。

BYSORT=NO 引数の場合は、BY ステートメントの前にすでにデータが並べ替えられている必要があります。並べ替えは、前の PROC SORT か、または BY 変数順序でのデータセットの作成によるものです。データセットが並べ替えられていない場合、エラーが発生します。BYSORT=NO の場合、グループ化されたデータはデータセット順序でアプリケーションに配信されます。インデックスは、BY 変数順序のオブザベーションの取得には使用されません。BYSORT=NO の場合、データセットオプション BYNOEQUALS=は無効になります。

LIBNAME ステートメントで BYSORT=オプションを指定する場合、PROC ステップまたは DATA ステップでの BYSORT=の指定によって上書きされる可能性があります。入力モードで開くために、LIBNAME ステートメントの BYSORT=NO を上書きするには、DATA ステップまたは PROC ステップで BYSORT=YES を設定

します。重要な点は、BYSORT=NO によって、エンジンがデータの並べ替え処理をしないように指示されることです。

MSGLEVEL=I SAS システムオプションを設定した場合、BYSORT=YES および IDXWHERE=データセットオプションを使用すると、次のメッセージが SAS ログに書き込まれます。

- IDXWHERE=YES で BY 変数にインデックスがある場合、テーブルの行の並べ替えにそのインデックスが使用されます。次のメッセージが SAS ログに書き込まれます。

  Note: BY ordering was produced by using an index for table *tablename*.

- IDXWHERE=NO または IDXWHERE=YES で、BY 変数にインデックスがない場合、SPD Engine で自動並べ替えが実行され、テーブルの行が並べ替えられます。次のメッセージが SAS ログに書き込まれます。

  Note: BY ordering was produced by performing an automatic sort on table *tablename*.

## 比較

BYSORT=データセットオプションで、BYSORT= LIBNAME ステートメントオプションが上書きされます。

## 例

### 例 1: BYSORT=YES を使用するデフォルトのグループフォーマット

```
libname growth spde 'SAS-library';
data growth.teens;
  input Name $ Sex $ Age Height Weight;
datalines;
Alfred M 14 69.0 112.5
Carol F 14 62.8 102.5
James M 13 57.3 83.0
Janet F 15 62.5 112.5
Judy F 14 64.3 90.0
Philip M 16 72.0 150.0
William M 15 66.5 112.0
;
proc print data=growth.teens; by sex;
run;
```

BYSORT=YES がデフォルトのため、PROC SORT を使用してデータを並べ替えていなくても、エラーは発生しません。

出力は次のとおりです。

**アウトプット 4.3** BYSORT=YES を使用するデフォルトのグループフォーマット

The SAS System

Sex=F

| Obs | Name | Age | Height | Weight |
|---|---|---|---|---|
| 2 | Carol | 14 | 62.8 | 102.5 |
| 4 | Janet | 15 | 62.5 | 112.5 |
| 5 | Judy | 14 | 64.3 | 90.0 |

Sex=M

| Obs | Name | Age | Height | Weight |
|---|---|---|---|---|
| 1 | Alfred | 14 | 69.0 | 112.5 |
| 3 | James | 13 | 57.3 | 83.0 |
| 6 | Philip | 16 | 72.0 | 150.0 |
| 7 | William | 15 | 66.5 | 112.0 |

### 例 2: BYSORT=NO

PROC PRINT ステートメントに BYSORT=NO を指定すると、自動並べ替えが抑制されている場合(BYSORT=NO)、必ずエラーが返されます。BY ステートメントの前に BY 変数でデータを並べ替える必要があります(PROC SORT などを使用)。

```
libname growth spde 'SAS-library';
proc print data=growth.teens (bysort=no);
by sex;
run;
ERROR: Data set GROWTH.TEENS is not sorted in ascending sequence.
      The current BY-group has Sex = M and the next BY-group has Sex = F.
NOTE: The SAS System stopped processing this step because of errors.
```

## COMPRESS=データセットオプション

SPD Engine データセットの作成時にディスク上で圧縮するように指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**デフォルト:** NO

**制限事項:** ENCRYPT=YES または ENCRYPT=AES と一緒には使用不可

**操作:** “IOBLOCKSIZE=データセットオプション” (73 ページ)
“PADCOMPRESS=データセットオプション” (77 ページ)

**エンジン:** SPD Engine のみ

## 構文

**COMPRESS=**NO | YES | CHAR | BINARY

### 必須引数

**NO**
データセットの圧縮は実行しません。

**YES | CHAR**
RLE (ランレングスエンコーディング)を使用して SPD Engine データセット内のデータをブロック単位で圧縮するように指定します。RLE では、反復実行される同じ文字(空白を含む)を 2 バイトまたは 3 バイトの表現に削減することでデータが圧縮されます。

**BINARY**
RDC (ロスデータ圧縮)を使用して SPD Engine データセット内のデータをブロック単位で圧縮するように指定します。RDC では、RLE とスライディングウィンドウ圧縮を組み合わせて反復バイトパターンをより効果的に表現することでファイルが圧縮されます。

*注:* これは、中容量から大容量(数百バイト以上)のブロックのバイナリデータ(文字変数と数値変数)を圧縮する場合に大きな効果を発揮する方法です。

## 詳細

COMPRESS=YES|BINARY|CHAR を指定した場合、SPD Engine では、データコンポーネントファイルが、作成時にブロックで圧縮されます。圧縮ブロックのサイズを指定するには、データセットの作成時に“IOBLOCKSIZE=データセットオプション” (73 ページ)を使用します。新しく圧縮したブロックに埋め込みを追加するには、データセットの作成または更新時に“PADCOMPRESS=データセットオプション” (77 ページ)を指定します。詳細については、“SPD Engine データセットの圧縮” (19 ページ)を参照してください。

SPD Engine では、ユーザー指定の圧縮はサポートされません。圧縮と暗号化の両方が行われたデフォルト Base SAS Engine データセットを移行する場合、暗号化は保持されますが、圧縮は解除されます。

CONTENTS プロシジャでは、圧縮設定が識別されます。データセットが圧縮される場合、PROC CONTENTS で圧縮についての情報が出力されます。次の例では、CONTENTS プロシジャ出力の圧縮情報フィールドについて説明します。

通常、COMPRESS=CHAR を指定すると、単一バイトの反復が存在する場合に適切な圧縮結果がもたらされ、COMPRESS=BINARY を指定すると、バイト列の反復が存在する場合に適切な圧縮結果がもたらされます。また、単一バイトの反復よりも、バイト列の反復を見つける方が労力を要します。たとえば、“例 1: COMPRESS=CHAR” (61 ページ)および“例 2: COMPRESS=BINARY” (62 ページ)を参照してください。

**アウトプット 4.4** PROC CONTENTS 圧縮情報セクション

| - Compressed Info | - |
|---|---|
| Number of compressed blocks | 202 |
| Raw data blocksize | 32736 |
| Number of blocks with overflow | 5 |
| Max overflow chain length | 3 |
| Block number for max chain | 80 |
| Min overflow area | 87 |
| Max overflow area | 181 |

圧縮ブロック数
: データの格納に必要な圧縮ブロック数。

生データブロックサイズ
: IOBLOCKSIZE=データセットオプションに指定したサイズから計算されたバイト単位の圧縮ブロックサイズ。これは、ブロックサイズ単位で取得するオブザベーション長の最大倍数です。

オーバーフローのあるブロック数
: より多くの領域を必要とする圧縮ブロック数。データが更新されて、圧縮された新ブロックが圧縮された旧ブロックよりも大きくなった場合、オーバーフローブロックフラグメントが作成されます。

オーバーフローチェーンの最大長
: 単一ブロックのオーバーフロー最大数。たとえば、圧縮ブロックが更新されて大きくなり、再度更新されてより大きなサイズになった場合、オーバーフローチェーンの最大長は 2 になります。

最大チェーンのブロック数
: 最大数のオーバーフローブロックを含むブロック数。

最小オーバーフロー領域
: オーバーフローに必要なディスク領域最小容量。

最大オーバーフロー領域
: オーバーフローに必要なディスク領域最大容量。

通常、圧縮ファイルへのアクセスには、より多くの処理時間を必要とします。読み取る前にファイルの圧縮解除が必要です。また、更新の場合は、ディスクに書き込む際に、もう一度圧縮する必要があります。

## 比較

COMPRESS=データセットオプションで、COMPRESS= LIBNAME ステートメントオプションおよび COMPRESS=システムオプションが上書きされます。

## 例

### 例 1: COMPRESS=CHAR

```
data mylib.CharRepeats(compress=char);
  length ca $ 200;
  do i=1 to 100000;
    ca='aaaaaaaaaaaaaaaaaaaaa';
    cb='bbbbbbbbbbbbbbbbbbbbb';
    cc='ccccccccccccccccccccc';
    output;
  end;
run;
```

次のメッセージが SAS ログに書き込まれます。

```
NOTE: Compressing data set MYLIB.CHARREPEATS decreased size by 88.55 percent.Compressed is 45
pages; un-compressed would require 393 pages.
```

### 例 2: COMPRESS=BINARY

```
data mylib.StringRepeats(compress=binary);
  length cabcd $ 200;
  do i=1 to 1000000;
    cabcd='abcdabcdabcdabcdabcdabcdabcdabcd';
    cefgh='efghefghefghefghefghefghefghefgh';
    cijkl='ijklijklijklijklijklijklijklijkl';
    output;
  end;
run;
```

次のメッセージが SAS ログに書き込まれます。

```
NOTE: Compressing data set MYLIB.STRINGREPEATS decreased size by 70.27 percent.Compressed is
1239 pages; un-compressed would require 4167 pages.
```

## ENCRYPT=データセットオプション

出力 SPD Engine データセットを暗号化するかどうかを指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**デフォルト:** NO

**制限事項:** 出力データセットにのみ使用
ENCRYPT=YES または ENCRYPT=AES は COMPRESS=と一緒には使用不可

### 構文

**ENCRYPT=** AES | NO | YES

### 構文の説明

**AES**
AES (Advanced Encryption Standard)アルゴリズムを使用してデータセットを暗号化します。AES は、Base SAS に含まれている SAS/SECURE ソフトウェアを使用することで、より強力な暗号化を提供します。ENCRYPT=AES を使用する場合、ENCRYPTKEY=データセットオプションを指定する必要があり

ます。詳細については、“ENCRYPTKEY=データセットオプション” (65 ページ)を参照してください。

注意 **ENCRYPT=AES を指定する場合、すべての ENCRYPTKEY=キー値を記録しておく必要があります。** ENCRYPTKEY=キー値を忘れた場合、データは失われます。SAS では ENCRYPTKEY=キー値の回復をサポートできません。次のメッセージが SAS ログに書き込まれます。

Note: If you lose or forget the ENCRYPTKEY= key value, there will be no way to open the file or recover the data.

**NO**
データセットを暗号化しません。

**YES**
SAS Proprietary アルゴリズムを使用してデータセットを暗号化します。この暗号化方法では、データセットに保存されたパスワードが使用されます。ENCRYPT=YES を指定した場合、少なくとも READ=または PW=データセットオプションを同時に指定する必要があります。この暗号化方法ではパスワードが使用されるため、暗号化されたデータセット上のパスワードは、データセットを再作成せずに変更することはできません。

**注意:**
**ENCRYPT=YES を指定する場合、すべてのパスワードを記録しておく必要があります。** パスワードを忘れた場合、SAS の支援なしではリセットができません。このプロセスには、時間がかかり、リソースが集中的に使用されます。

## 詳細

SPD Engine では、暗号化と圧縮は相互に排他的です。

SPD Engine データセットを作成する際に暗号化と圧縮を両方とも行うことはできません。ENCRYPT=YES または ENCRYPT=AES と、COMPRESS=データセットオプションまたは LIBNAME オプションを同時に使用すると、次のエラーが発生します。

ERROR: The data set was not created because compression and encryption cannot both be specified.

圧縮化され暗号化された Base SAS データセットは、SPD Engine にコピーできません。

ENCRYPT=YES を使用する場合、次の規則が適用されます。

- 暗号化データセットをコピーするには、出力エンジンで暗号化がサポートされている必要があります。サポートされていない場合、データセットはコピーされません。
- データセットが暗号化される場合、関連するインデックスファイルおよびメタデータファイルもすべて暗号化されます。
- 暗号化には、圧縮とおおよそ同じ量の CPU リソースが必要です。
- PROC CPORT は、SAS Proprietary で暗号化されたデータセットに対しては使用できません。

ENCRYPT=AES を使用する場合、次の規則が適用されます。

- AES 暗号化でデータセットを作成する場合は、ENCRYPTKEY=データセットオプションを使用する必要があります。

- AES で暗号化されたデータセットをコピーするには、出力エンジンで AES 暗号化がサポートされている必要があります。サポートされていない場合、データセットはコピーされません。
- データセットが暗号化される場合、関連するインデックスファイルもすべて暗号化されます。
- SAS 9.4 より前のリリースでは、AES で暗号化されたデータセットは使用できません。
- AES 暗号化を利用する場合、Base SAS に含まれている SAS/SECURE ソフトウェアを使用します。

AES で暗号化されたデータセットでは ENCRYPTKEY=キー値を変更できません。これを変更するにはデータセットの再作成が必要となります。

## 例

### 例 1: ENCRYPT=YES オプションの使用

次の例では、SAS Proprietary アルゴリズムを使用しています。

```
libname depta spde 'SAS-library';
data salary(encrypt=yes read=green);
 input name $ yrsal bonuspct;
datalines;
Muriel    34567  3.2
Bjorn     74644  2.5
Freda     38755  4.1
Benny     29855  3.5
Agnetha   70998  4.1
;
```

このデータセットを使用するには、READ パスワードを指定します。

```
proc contents data=salary(read=green);
run;
```

### 例 2: ENCRYPT=AES オプションの使用

次の例では、AES アルゴリズムを使用しています。

```
data salary(encrypt=aes encryptkey=green);
  input name $ yrsal bonuspct;
  datalines;
Muriel    34567  3.2
Bjorn     74644  2.5
Freda     38755  4.1
Benny     29855  3.5
Agnetha   70998  4.1
```

このデータセットを使用するには、ENCRYPTKEY=キー値を指定します。

```
proc contents data=salary(encryptkey=green);
run;
```

### 例 3: AES で暗号化されたデータセットのコピー

次の 2 つの例では、ENCRYPTKEY=データセットオプションと COPY プロシジャを使用しています。

```
PROC COPY IN=inlib OUT=outlib ENCRYPTKEY=secret;
   SELECT abc (ENCRYPTKEY=secreta) DEF(ENCRYPTKEY=secretb)...


PROC COPY IN=inlib OUT=outlib;
   SELECT abc (ENCRYPTKEY=secreta) DEF(ENCRYPTKEY=secretb)...
```

## ENCRYPTKEY=データセットオプション

AES 暗号化のキー値を指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**範囲:** 1〜64 バイト

**制限事項:** SAS 9.4 以降でのみ使用可能
AES で暗号化されたデータセットでのみ使用可能

### 構文

**ENCRYPTKEY=***key-value*

### 構文の説明

***key-value***
暗号化のキー値を割り当てます。ENCRYPTKEY=データセットオプションは、ENCRYPT=AES と一緒に使用する必要があります。キー値の長さは最大で 64 バイトです。ENCRYPTKEY=キー値は、次のルールに従い、必要に応じて引用符を付けて作成します。

引用符なし

- 英数字文字とアンダースコアのみ
- 最大 64 バイト
- 大文字と小文字の英字のみ
- 先頭は必ず英字にする
- 空白を含まない
- 大文字小文字を区別しない

encryptkey=*key-value*
encryptkey=*key-value1*

一重引用符

- 英数字文字、特殊文字および DBCS 文字
- 最大 64 バイト
- 大文字と小文字の英字のみ
- 大文字小文字を区別する

encryptkey='*key-value*'
encryptkey='*1234*#mykey*'

二重引用符

- 英数字文字、特殊文字および DBCS 文字
- 最大 64 バイト
- 大文字と小文字の英字のみ
- マクロ解決が可能
- 大文字小文字を区別する

```
encryptkey="key-value"
encryptkey="1234*#mykey"
%let mykey=abcdefghi12;
encryptkey=&key-value
```

ENCRYPTKEY=キー値で DBCS 文字を使用する場合、文字列が UTF-8 エンコーディングに変換された後、その文字列には 64 バイトの制限が適用されます。DBCS によるキー値の長さを計算するには、次の DATA ステップを使用します。

```
data _null_;
  key=length(unicodec('key-value','UTF8'));
  put 'key length=' key;
run;
```

**注** AES で暗号化されたデータセットでは ENCRYPTKEY=キー値を変更できません。これを変更するにはデータセットの再作成が必要となります。

## 詳細

**注意:**

**ENCRYPTKEY=キー値は記録しておく必要があります。** ENCRYPTKEY=キー値を忘れた場合、データは失われます。SAS では ENCRYPTKEY=キー値の回復をサポートできません。

AES 暗号化を使用して SPD Engine データセットを作成する場合や同データセットにアクセスする場合、ENCRYPTKEY=データセットオプションを使用する必要があります。

ENCRYPTKEY=データセットオプションは、データセットを削除や置換から保護するものではありません。次のいずれかの方法を使用すれば、キー値を指定せずに暗号化されたデータセットを削除できます。

- PROC DATASETS の KILL オプション
- PROC SQL の DROP ステートメント
- DELETE プロシジャ

ENCRYPTKEY=データセットオプションは、データセットの中身に対してのみアクセスを拒否するものです。データセットを削除や置換から保護するには、そのデータセットに ALTER= パスワードを含める必要があります。

AES で暗号化されたデータセットをコピーする場合、ENCRYPTKEY=キー値を指定する必要があります。SELECT ステートメントで、データセット名の後にキー値を指定します。SELECT の使用例を次に示します。

```
copy in=OldLib out=NewLib;
  select salary(encryptkey=key-value);
run;
```

ENCRYPTKEY=キー値で保護されているデータセットを DATASETS プロシジャで取り扱う場合、AGE、APPEND、CONTENTS、MODIFY の各ステートメントでキー値を指定できます。ENCRYPTKEY=データセットオプションは、SAS データセット名の後にかっこで囲んで指定するか、またはスラッシュの後に指定します。

マクロ変数を ENCRYPTKEY=キー値として使用することもできます。マクロ変数を使用する場合、二重引用符を使用する必要があります。マクロ変数を定義する例を次に示します。

```
%let secret=MyValue;
```

次の例では、このマクロ変数を ENCRYPTKEY=キー値として使用しています。

```
data my.dsname(encrypt=aes encryptkey="&secret");
```

このマクロ変数を ENCRYPTKEY=キー値として指定する場合、そのマクロ変数を二重引用符で囲む必要があります。二重引用符を使用しない場合、予期せぬ結果が引き起こされることがあります。

### 例: ENCRYPTKEY=データセットオプションの使用

次の例では ENCRYPT=AES オプションを使用しています。

```
data spdelib.salary(encrypt=aes encryptkey=green);
  input name $ yrsal bonuspct;
  datalines;
Muriel   34567  3.2
Bjorn    74644  2.5
Freda    38755  4.1
Benny    29855  3.5
Agnetha  70998  4.1
```

このデータセットを使用するには、ENCRYPTKEY=キー値を指定します。

```
proc contents data=spdelib.salary(encryptkey=green);
run;
```

## ENDOBS=データセットオプション

ユーザー定義のオブザベーション処理範囲における最後のオブザベーション番号を指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**デフォルト:** データセット内の最後のオブザベーション

**制限事項:** ENDOBS=は入力データセットにのみ使用

OBS=システムまたはデータセットオプション、あるいは FIRSTOBS=システムおよびデータセットオプションと一緒には使用不可

**操作:** “ENDOBS= LIBNAME ステートメントオプション” (35 ページ)
“STARTOBS= LIBNAME ステートメントオプション” (44 ページ)
“STARTOBS=データセットオプション” (80 ページ)

**エンジン:** SPD Engine のみ

### 構文

**ENDOBS=*n***

### 必須引数

***n***
終了オブザベーションの番号。

## 詳細

### オブザベーション範囲の指定

デフォルトでは、SPD Engine で、STARTOBS=オプションまたは ENDOBS=オプションを使用してオブザベーションの範囲を指定しない限り、データセット全体のすべてのオブザベーションが処理されます。STARTOBS=オプションを ENDOBS=オプションなしで使用した場合、ENDOBS=の暗黙値はデータセットの終わりになります。両方のオプションを一緒に使用する場合、ENDOBS=値を STARTOBS=値よりも大きくする必要があります。

SPD Engine における ENDOBS=データセットオプションは、デフォルトの Base SAS Engine における OBS=データセットオプションと同様に機能します。唯一の違いは、ENDOBS=を WHERE 式で使用した場合の動作です。

### WHERE 式での ENDOBS=の使用

ENDOBS=を WHERE 式で使用する場合、ENDOBS=値は、返されるオブザベーションの数ではなく、最後に処理するオブザベーションを表します。次の例でその違いを示します。

*注:* OBS=システムオプションおよび OBS=データセットオプションは、STARTOBS=もしくは ENDOBS=データセットまたは LIBNAME オプションと一緒には使用できません。

## 比較

ENDOBS=データセットオプションで、ENDOBS= LIBNAME ステートメントオプションが上書きされます。

## 例

### 例 1: ENDOBS=データセットオプションの使用

SPD Engine で ENDOBS=5 オプションを指定してデータセットを作成および処理します。オブザベーション番号 5 で終わるデータセットに WHERE 式が適用されます。PRINT プロシジャでは 4 つのオブザベーションが印刷されます。これは WHERE 式に適合するオブザベーションです。

```
libname growth spde 'SAS-library';
data growth.teens;
  input Name $ Sex $ Age Height Weight;
  list;
datalines;
Alfred M 14 69.0 112.5
Carol F 14 62.8 102.5
James M 13 57.3 83.0
Janet F 15 62.5 112.5
Judy F 14 64.3 90.0
Philip M 16 72.0 150.0
Zeke M 14 71.1 105.0
Alice F 14 65.1 91.0
```

```
William M 15 66.5 112.0
;
proc print data=growth.teens (endobs=5);
  where age >13;
  title 'WHERE age > 13 using SPD Engine';
run;
```

**アウトプット 4.5** ENDOBS=

WHERE age > 13 using SPD Engine

| Obs | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 1 | Alfred | M | 14 | 69.0 | 112.5 |
| 2 | Carol | F | 14 | 62.8 | 102.5 |
| 4 | Janet | F | 15 | 62.5 | 112.5 |
| 5 | Judy | F | 14 | 64.3 | 90.0 |

## 例 2: SPD Engine での OBS=

OBS=5 を指定して同じデータセットを処理します。PROC PRINT では 5 つのオブザベーションが印刷されます。これは WHERE 式に適合するすべてのオブザベーションで、5 番目の適合オブザベーションで終了します。

```
libname growth spde 'SAS-library';
data growth.teens;
  input Name $ Sex $ Age Height Weight;
  list;
datalines;
Alfred M 14 69.0 112.5
Carol F 14 62.8 102.5
James M 13 57.3 83.0
Janet F 15 62.5 112.5
Judy F 14 64.3 90.0
Philip M 16 72.0 150.0
Zeke M 14 71.1 105.1
Alice F 14 65.1 91.0
William M 15 66.5 112.0
;
proc print data=growth.teens (obs=5);
  where age >13;
  title 'WHERE age > 13 using V9';
run;
```

**アウトプット 4.6** OBS=

WHERE age > 13 using V9

| Obs | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 1 | Alfred | M | 14 | 69.0 | 112.5 |
| 2 | Carol | F | 14 | 62.8 | 102.5 |
| 4 | Janet | F | 15 | 62.5 | 112.5 |
| 5 | Judy | F | 14 | 64.3 | 90.0 |
| 6 | Philip | M | 16 | 72.0 | 150.0 |

## IDXBY=データセットオプション

SPD Engine での BY ステートメント処理時にインデックスを使用するかどうかを指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**デフォルト:** YES

**エンジン:** SPD Engine のみ

### 構文

**IDXBY=**YES | NO

#### 必須引数

**YES**
BY ステートメントのインデックス変数を処理するときにインデックスを使用します。

*注:* BY ステートメントに 2 つ以上の変数または DESCENDING オプションが指定されている場合は、IDXBY=YES でもインデックスは使用されません。

**NO**
BY ステートメントのインデックス変数を処理するときにインデックスを使用しません。

注 IDXBY=NO の場合、BY ステートメントの処理時に自動並べ替えが実行されます。

### 詳細

IDXBY=データセットオプションを使用する場合は、BYSORT=YES オプションを使用し、BY 変数にインデックスを付けるようにしてください。

場合によっては、データを自動的に並べ替えると、SPD Engine のパフォーマンスが向上することがあります。自動並べ替えを使用するには、BYSORT=YES を設定し、IDXBY=NO を指定する必要があります。

SAS システムオプション MSGLEVEL=I を設定し、BY 処理情報が SAS ログに書き込まれるようにします。IDXBY=データセットオプションおよび BYSORT=YES オプションを使用すると、SAS ログに次のメッセージが書き込まれます。

- IDXBY=YES で BY 変数にインデックスがある場合、テーブルの行の並べ替えにそのインデックスが使用されます。次のメッセージが SAS ログに書き込まれます。

  NOTE: BY ordering was produced by using an index for
  table *tablename*.

- IDXBY=NO の場合、次のメッセージが SAS ログに書き込まれます。

  NOTE: BY ordering was produced by performing an automatic sort
  on table *tablename*.

## 比較

IDXBY=データセットオプションで、IDXBY= LIBNAME ステートメントオプションが上書きされます。

## 例

### 例 1: IDXBY=NO データセットオプションの使用

```
options msglevel=i;
proc means data=permdata.customer(IDXBY=no);
  by sales;
  by state;
run;
```

次のメッセージが SAS ログに書き込まれます。

```
NOTE: BY ordering was produced by performing an automatic sort
    on table PERMDATA.customer.
NOTE: There were 2981 observations read from the data set
    PERMDATA.CUSTOMER.
```

### 例 2: IDXBY=YES データセットオプションの使用

```
proc means data=permdata.customer(IDXBY=yes);
  var sales;
  by state;
run;
```

次のメッセージが SAS ログに書き込まれます。

```
NOTE: BY ordering was produced by using an index for table
    PERMDATA.customer.
NOTE: There were 2981 observations read from the data set
    PERMDATA.CUSTOMER.
```

# IDXWHERE=データセットオプション

SPD Engine での WHERE 式処理時にインデックスを使用するかどうかを指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**デフォルト:** YES

**制限事項:** WHERENOINDEX=オプションは IDXWHERE=NO オプションと一緒には使用不可

**エンジン:** SPD Engine のみ

## 構文

**IDXWHERE=**YES | NO

### 必須引数

**YES**
WHERE 式の処理時にインデックスを使用します。

**NO**
WHERE 式の処理時にインデックスを無視します。

制限事項 IDXWHERE=NO オプションと WHERENOINDEX=オプションを一緒に使用することはできません。

## 詳細

IDXWHERE=は、WHINIT と呼ばれる、SPD Engine の WHERE 式計画ソフトウェアで使用されます。WHINIT では、さまざまなアプリケーションにおける WHERE 処理でのインデックス使用のパフォーマンスがテストされます。SAS システムオプション MSGLEVEL=I を設定し、WHERE 処理情報が SAS ログに出力されるようにします。

IDXWHERE=データセットオプションおよび BYSORT=YES オプションを使用すると、SAS ログに次のメッセージが書き込まれます。

- IDXWHERE=YES で BY 変数にインデックスがある場合、テーブルの行の並べ替えにそのインデックスが使用されます。次のメッセージが SAS ログに書き込まれます。

  Note: BY ordering was produced by using an index for table *tablename*.

- IDXWHERE=NO または IDXWHERE=YES で、BY 変数にインデックスがない場合、SPD Engine で自動並べ替えが実行され、テーブルの行が並べ替えられます。次のメッセージが SAS ログに書き込まれます。

  Note: BY ordering was produced by performing an automatic sort on table *tablename*.

SPD Engine は、WHINIT (規則に基づく WHERE 式プランナ)を使用して、最もクエリに適した評価方法を選択します。SAS システムオプション MSGLEVEL=I によって、WHINIT メッセージが SAS ログに出力されます。これは、1 つ以上のインデックスがクエリで使用されるかどうかを決定するのに役立ちます。WHINIT の詳細については、“SPDEWHEVAL=システムオプション” (100 ページ)を参照してください。

*注: WHERE ステートメントと BY ステートメントを一緒に使用する場合は、インデックスの使用を任意で抑制しないでください。*WHERE ステートメントでオブザベーションのフィルタ処理、BY ステートメントでオブザベーションの並べ替えを行います。WHERE ステートメントに適合する、フィルタ処理されたオブザベーションが、並列 WHERE 式評価の一部として並べ替えステップに直接取り込まれます。結果として、最終的に並べ替えられたオブザベーションセットが作成されます。WHERE 処理でのインデックス使用によって、並べ替えステップでのフィルタ処理および取り込みのパフォーマンスが大幅に向上します。

## 例

### 例 1: IDXWHERE=NO で WHINIT ログ出力を使用

この例では、IDXWHERE=NO が指定されたため、WHERE 式で評価方法 2 が使用されます。

**ログ 4.3** IDXWHERE=NO

```
34 options msglevel=i; 35 proc means data=permdata.customer(idxwhere=no); 36 var sales; 37 where state="CA"; 38 run; whinit: WHERE (sstate='CA') whinit returns: ALL EVAL2 NOTE: There were 2981 observations read from the data set PERMDATA.CUSTOMER.WHERE state='CA';
```

### 例 2: IDXWHERE=YES で WHINIT ログ出力を使用

この例では、IDXWHERE=YES が指定されたため、評価方法 1 が使用されました。

**ログ 4.4** IDXWHERE=YES

```
39 proc means data=permdata.customer(idxwhere=yes); 40 var sales; 41 where state="CA"; 42 run; whinit: WHERE (sstate='CA') -- whinit: SBM-INDEX STATE uses 45% of segs (WITHIN maxsegratio 75%) whinit returns: ALL EVAL1(w/SEGLIST) NOTE: There were 2981 observations read from the data set PERMDATA.CUSTOMER.WHERE state='CA';
```

# IOBLOCKSIZE=データセットオプション

I/O 操作で使用するオブザベーションブロックのサイズをバイト単位で指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**デフォルト:** 1,048,576 バイト(1 MB)

**範囲:** 最小ブロックサイズは 32,768 バイトです。最大ブロックサイズは SPD Engine データパーティションファイルの半分のサイズです。

**制限事項:** データセットの読み取り時に、ブロックサイズがパフォーマンスに大きな影響を及ぼすことがあります。データの大部分を取得する場合、ブロックサイズが大きい方がパフォーマンスが向上します。ただし、WHERE 処理などで、データのサブセットを取得する場合は、ブロックサイズが小さいほうがパフォーマンスが上がります。

**エンジン:** SPD Engine のみ

## 構文

**IOBLOCKSIZE=***n*

### 必須引数

***n***
オブザベーションブロックのバイト単位でのサイズです。

## 詳細

I/O ブロックサイズによって、I/O 操作で物理的に一緒に転送されるデータ量が決定されます。ブロックサイズが大きいほど、I/O は少なくなります。SPD Engine では、メモリ内のブロックを使用して、データコンポーネントファイル

での読み書き対象のオブザベーションを収集します。IOBLOCKSIZE=データセットオプションでは、ブロックのサイズが指定されます。(実際のサイズは、*n* バイトの指定サイズに収まるオブザベーション最大数に対応するように計算されます。したがって、実際のサイズはオブザベーションの長さの倍数になります)。

ブロックサイズは、圧縮データセット、非圧縮データセット、および暗号化データセットに対する I/O 操作に影響します。ただし、その影響はさまざまで、I/O 操作に依存します。

- 圧縮データセットの場合、ブロックサイズによって、一緒に圧縮されるオブザベーションの数が決定されます。これにより、読み取り操作と書き込み操作の両方について物理的に転送されるデータの量が決定されます。ブロックサイズはファイルの永久属性です。異なるブロックサイズを指定するには、データセットを新しいデータセットにコピーし、出力ファイルに対して新しいブロックサイズを指定する必要があります。圧縮データセットの場合、読み取り操作と書き込み操作のどちらでもブロックサイズが大きい方がパフォーマンスを向上させられます。
- 暗号化データセットの場合、ブロックサイズはファイルの永久属性です。
- 非圧縮データセットの場合、ブロックサイズによって、ディスクからメモリへのデータ読み取りに使用されるブロックのサイズが決定されます。ブロックサイズは、データのディスクへの書き込み時には、影響はありません。非圧縮データセットの場合、ブロックサイズはファイルの永久属性ではありません。つまり、実行する読み取り操作に基づいて異なるブロックサイズを指定できます。たとえば、ランダムに分散されたデータの読み取りや、データのサブセットの読み取りでは、ブロックサイズを小さくする必要があります。これは、小さいブロックにアクセスする方が、大きいブロックにアクセスするよりも速いためです。それとは対照的に、均等もしくは順番に分散されたデータ、またはデータセット全体のスキャンが必要なデータの読み取りは、ブロックサイズが大きいほうがより適切に動作します。

### 比較

IOBLOCKSIZE=データセットオプションで、IOBLOCKSIZE= LIBNAME ステートメントオプションが上書きされます。

### 例: IOBLOCKSIZE=の使用

```
/*IOBLOCKSIZE set to 64K */
data sport.maillist(ioblocksize=65536);
/*IOBLOCKSIZE set to 32K */
data sport.maillist(ioblocksize=32768 compress=yes);
```

## LISTFILES=データセットオプション

CONTENTS プロシジャで、SPD Engine データセットのコンポーネントファイルすべての完全パス名をリストするかどうかを指定します。

**該当要素:** PROC CONTENTS のみ

**デフォルト:** NO

**エンジン:** SPD Engine のみ

## 構文

**LISTFILES=**YES | NO

### 必須引数

**YES**
SPD Engine データセットのコンポーネントファイルすべての完全パス名をリストします。

**NO**
パス名をリストしません。

## 詳細

LISTFILES=データセットオプションは SPD Engine でのみ使用されます。SPD Engine データセットのコンポーネントファイルすべての完全パス名をリストするには、CONTENTS プロシジャを使用します。

## 例: LISTFILES オプション

```
proc contents data=company.depts (listfiles=yes);
```

次の CONTENTS プロシジャ出力には、すべてのコンポーネントファイルの完全パス名が表示されています。

**アウトプット 4.7** CONTENTS プロシジャ—出力セクション 1

The SAS System

The CONTENTS Procedure

| | | | |
|---|---|---|---|
| Data Set Name | COMPANY.DEPTS | Observations | 285120 |
| Member Type | DATA | Variables | 8 |
| Engine | SPDE | Indexes | 1 |
| Created | Tuesday, December 14, 2010 04:41:29 PM | Observation Length | 152 |
| Last Modified | Tuesday, December 14, 2010 04:47:15 PM | Deleted Observations | 0 |
| Protection | | Compressed | NO |
| Data Set Type | | Sorted | NO |
| Label | | | |
| Data Representation | WINDOWS_64 | | |
| Encoding | wlatin1 Western (Windows) | | |

**アウトプット 4.8** CONTENTS プロシジャー出力セクション 2

| Engine/Host Dependent Information | |
|---|---|
| Blocking Factor (obs/block) | 431 |
| Data Partsize | 10481920 |
| - Alphabetic List of Index Info | - |
| Index | JOB1 |
| KeyValue (Min) | ACCOUNTANT |
| KeyValue (Max) | TRANSLATOR |
| Number of discrete values | 29 |
| - Metadata Files | - |
| d:\main\depts.mdf.0.0.0.spds9 | - |
| - Data Files | - |
| d:\data\depts.dpf.03bf03f7.0.161.spds9 | - |
| d:\data\depts.dpf.03bf03f7.1.161.spds9 | - |
| d:\data\depts.dpf.03bf03f7.2.161.spds9 | - |
| d:\data\depts.dpf.03bf03f7.3.161.spds9 | - |
| d:\data\depts.dpf.03bf03f7.4.161.spds9 | - |
| - Index Files | - |
| d:\indexes\depts.idxjob1.03bf03f7.0.161.spds9 | - |
| d:\indexes\depts.hbxjob1.03bf03f7.0.161.spds9 | - |

**アウトプット 4.9** CONTENTS プロシジャー出力セクション 3

Alphabetic List of Variables and Attributes

| # | Variable | Type | Len |
|---|---|---|---|
| 4 | DEPTHEAD | Char | 15 |
| 7 | JOB1 | Char | 15 |
| 2 | LEVEL1 | Char | 16 |
| 1 | LEVEL2 | Char | 13 |
| 5 | LEVEL3 | Char | 20 |
| 6 | LEVEL4 | Char | 30 |
| 3 | LEVEL5 | Char | 30 |
| 8 | N | Num | 8 |

Alphabetic List of Indexes and Attributes

| # | Index | # of Unique Values |
|---|---|---|
| 1 | JOB1 | 29 |

## PADCOMPRESS=データセットオプション

OUTPUT または UPDATE モードで開かれるデータセットの圧縮ブロックに追加するバイト数を指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**デフォルト:** 0

**操作:** “COMPRESS=データセットオプション” (59 ページ)
“IOBLOCKSIZE=データセットオプション” (73 ページ)

**エンジン:** SPD Engine のみ

### 構文

**PADCOMPRESS=*n***

### 必須引数

***n***
追加するバイト数です。

### 詳細

圧縮 SPD Engine データセットはディスク上で複数ブロックの領域を占有します。ブロックのサイズは、データセットの作成時に指定された IOBLOCKSIZE=データセットオプションから得られます。データセットが更新されると、その更新を保持するために新しいブロックフラグメントの作成が必要になる可能性があります。更新を重ねると新しいフラグメントが作成される可能性があり、その結果、データセットの読み取りに必要な I/O 操作の回数が増加します。

データセットへの更新が多いことが予想される状況では、ブロック埋め込みを増加させることによって断片化を最小限に抑えられます。ただし、データセットを更新しない場合は、埋め込みの追加によって領域を無駄に消費する可能性があります。

すべての圧縮ブロックの埋め込みコストと、一部の圧縮ブロックで発生の可能性がある断片化のコストとを比較検討する必要があります。

データセットの作成または更新時に PADCOMPRESS=データセットオプションを指定すると、ブロックをディスクに書き込み直す際、すべてのブロックに領域が追加されます。PADCOMPRESS=設定は、データセットのメタデータでは保持されません。

## PARTSIZE=データセットオプション

データコンポーネントパーティションに指定可能な最大サイズ(MB、GB または TB 単位)を指定します。値は SPD Engine データセットの作成時に指定されます。このサイズは固定サイズです。この指定はデータコンポーネントファイルにのみ適用されます。

**該当要素:** DATA ステップおよび PROC ステップ

**使用要素:** MINPARTSIZE=システムオプション

**デフォルト:** 128 MB

**操作:** “DATAPATH= LIBNAME ステートメントオプション” (34 ページ)

**エンジン:** SPD Engine のみ

### 構文

**PARTSIZE=***n* | *n*M | *n*G | *n*T

#### 必須引数

***n* | *n*M | *n*G | *n*T**
MB、GB または TB でのパーティションのサイズです。M、G または T なしで *n* を指定した場合のデフォルトは MB です。たとえば、PARTSIZE=128 と PARTSIZE=128M は同じ意味です。最大値は、8,796,093,022,207MB です。

制限事項 この制限は、オペレーティングシステム z/OS、Linux SLES 9 x86 および Windows ファミリの 32 ビットホストにのみ適用されます。SAS 9.3 でパーティションサイズが 2GB 以上のデータセットを作成した場合、SAS 9.2 以前のいずれのバージョンの SPD Engine でもそのデータセットを開くことはできません。次のエラーメッセージが SAS ログに書き込まれます。**ERROR: Unable to open data file because its data representation differs from the SAS session data representation.**

## 詳細

複数のパーティションでは、データを並列処理で読み取る必要があります。オプション PARTSIZE=では、SPD Engine データファイルに指定サイズでパーティションが切られます。実際のパーティションサイズは、*n*MB、GB または TB の指定サイズに収まるオブザベーション最大数に対応するように計算されます。オブザベーションの長さが 65K より大きいテーブルの場合、指定した PARTSIZE=と実際のパーティションサイズが一致しない可能性があります。この数字を一致させるには、PARTSIZE=に、32 およびオブザベーションの長さの倍数を指定します。

SPD Engine データセットのデータ部分を固定サイズのファイルに分割(パーティション)することによって、ソフトウェアで一部の操作に高度なスケーラビリティが導入されます。SPD Engine では、スレッドを並列起動できます(たとえば、WHERE 評価に対して 1 パーティションにつき 1 スレッドまで)。また、別々のデータパーティションによって、スレッド間のファイルアクセス競合のオーバーヘッドなしでデータを処理できます。各パーティションが 1 ファイルであるため、小さなパーティションサイズの代償として、オブザベーションを格納するためにファイル(UNIX i ノードなど)の数の増加が必要になります。

PARTSIZE=を使用したスケーラビリティ制限は、DATAPATH=オプションで指定したファイルシステムを、どのように構成し、どのように複数のストライプボリュームに分散させるのかによって決まります。(個々のボリュームのストライピング構成を、ディスクアレイの複数のディスクコントローラまたは SCSI チャネルに分散させる必要があります。)ハードウェアレベルでの構成の目標は、データ取得時の並列処理を最大限にすることです。ディスクストライピングの詳細については、Scalability and Performance の“SPD Engine”の下に示されている“I/O Setup and Validation” (http://support.sas.com/rnd/scalability)を参照してください。

PARTSIZE=の指定は SPD Engine システムオプション MINPARTSIZE=によって制限されます。このシステムオプションは通常システム管理者によって保守されます。MINPARTSIZE=を使用すると、経験不足のユーザーが任意に小さなパーティションを作成し、その結果、多数のデータファイルが生成されるようなことは起こらなくなります。

パーティションサイズによって、データセット全体のスキャンを必要とする並列操作の多くの作業単位が決定されます。ただし、パーティションの増加が必ずしも処理の迅速化を意味するとは限りません。この折り合いをつけるには、データセットの格納に必要な物理ファイル(パーティション)の増加数と、パーティションの増加によって並列処理が可能になる作業量とのバランスをとる必要があります。パーティションが多くなると、データセットを処理するために開くファイルも多くなりますが、各パーティションのオブザベーション数は少なくなります。一般的なルールでは、パーティションは、1 データパスにつき 10 以下、1CPU につき 3 から 4 までとなります。(一部のオペレーティングシステムでは、使用可能な開いたファイルの数に制限がかかることもあります。)

新しい SPD Engine データセットの適切なパーティションサイズを決定するには、次の点に注意する必要があります。

- データに対して実行するアプリケーションの種類
- データ量はどの程度か
- アプリケーションで使用可能な CPU はいくつあるか
- パーティションの格納に使用可能なディスクはどれか
- そのディスクと CPU との関係

たとえば、各 CPU の制御するディスクが 1 つだけの場合、適切なパーティションサイズは、各ディスクにほぼ同じ量のデータが含まれるサイズです。各 CPU

の制御するディスクが 2 つの場合、適切なパーティションサイズは、負荷のバランスがとれているサイズです。各 CPU で、ほぼ同じ量の作業が行われます。

*注:* データセットの PARTSIZE=値は、データセットの作成後は変更できません。PARTSIZE=を変更するには、データセットを再作成し、LIBNAME ステートメント内か、またはその新しい(出力)データセットに対して、異なる PARTSIZE=値を指定する必要があります。

### 比較

PARTSIZE=データセットオプションで、PARTSIZE= LIBNAME ステートメントオプションが上書きされます。

### 例: PROC SQL の使用

100GB のデータと 8 つのディスクがあり、1 ディスクにつき 12.5GB を格納できるとします。パーティションは 1 ディスクにつき 3 つから 4 つまでが最適です。3.125GB のパーティションサイズが適切です。そのため、PARTSIZE=3200M を指定します。

```
data salecent.sw (partsize=3200m);
```

データ量は同じで、1 年以内にデータ量が 2 倍になることが予想されるとします。その場合、同じ PARTSIZE=を指定して 1 ディスクにつきおよそ 7 パーティションにするか、または PARTSIZE=を 5000M に増やして 1 ディスクにつき 5 パーティションにします。

## STARTOBS=データセットオプション

ユーザー定義のオブザベーション処理範囲における開始オブザベーション番号を指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**デフォルト:** データセット内の最初のオブザベーション

**制限事項:** STARTOBS=は入力データセットにのみ使用

OBS=システムまたはデータセットオプション、あるいは FIRSTOBS=システムおよびデータセットオプションと一緒には使用不可

**操作:** “STARTOBS= LIBNAME ステートメントオプション” (44 ページ)
“ENDOBS= LIBNAME ステートメントオプション” (35 ページ)
“ENDOBS=データセットオプション” (67 ページ)

**エンジン:** SPD Engine のみ

### 構文

**STARTOBS=*n***

#### 必須引数

***n***
開始オブザベーションの番号。

## 詳細

### オブザベーション範囲の指定

デフォルトでは、SPD Engine で、STARTOBS=オプションと ENDOBS=オプションを使用してオブザベーションの範囲を指定しない限り、データセット全体のすべてのオブザベーションが処理されます。ENDOBS=オプションを STARTOBS=オプションなしで使用した場合、STARTOBS=の暗黙値は 1 になります。両方のオプションを一緒に使用する場合、STARTOBS=値を ENDOBS=値よりも小さくする必要があります。

SPD Engine における FIRSTOBS=データセットオプションは、デフォルトの Base SAS Engine における FIRSTOBS=データセットオプションと同様に機能します。唯一の違いは、STARTOBS=を WHERE 式で使用した場合の動作です。

*注:* FIRSTOBS= SAS データセットオプションは、SPD Engine ではサポートされていません。OBS=システムオプションおよび OBS=データセットオプションは、STARTOBS=もしくは ENDOBS=データセットまたは LIBNAME オプションと一緒には使用できません。

### WHERE 式での STARTOBS=の使用

STARTOBS=を WHERE 式で使用する場合、STARTOBS=値は、WHERE 式を適用する最初のオブザベーションを表します。この値をデフォルト Base SAS Engine データセットオプション FIRSTOBS=と比較します。FIRSTOBS=では、WHERE 式に適合するデータのサブセット内の開始オブザベーション番号が指定されます。

## 比較

STARTOBS=データセットオプションで、STARTOBS= LIBNAME ステートメントオプションが上書きされます。

## 例

### 例 1: SPD Engine での STARTOBS=

SPD Engine で STARTOBS=5 を指定してデータセットを作成および処理します。オブザベーション番号 5 で始まるデータセットに WHERE 式が適用されます。PRINT プロシジャでは 6 つのオブザベーションが印刷されます。これは WHERE 式に適合するオブザベーションです。

```
libname growth spde 'SAS-library';
data growth.teens;
  input Name $ Sex $ Age Height Weight;
  list;
datalines;
Alfred M 14 69.0 112.5
Carol F 14 62.8 102.5
James M 13 57.3 83.0
Janet F 15 62.5 112.5
Judy F 14 64.3 90.0
Philip M 16 72.0 150.0
Zeke M 14 71.1 105.1
Alice F 14 65.1 91.0
William M 15 66.5 112.0
Mike M 16 67.0 105.1
```

```
;
proc print data=growth.teens (startobs=5);
  where age >13;
  title 'WHERE age>13 using SPD Engine';
run;
```

**アウトプット 4.10** STARTOBS=

**WHERE age > 13 using SPD Engine**

| Obs | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 5 | Judy | F | 14 | 64.3 | 90.0 |
| 6 | Philip | M | 16 | 72.0 | 150.0 |
| 7 | Zeke | M | 14 | 71.1 | 105.1 |
| 8 | Alice | F | 14 | 65.1 | 91.0 |
| 9 | William | M | 15 | 66.5 | 112.0 |
| 10 | Mike | M | 16 | 67.0 | 105.1 |

### 例 2: デフォルト Base SAS Engine での FIRSTOBS=

デフォルト Base SAS Engine で FIRSTOBS=5 オプションを指定して同じデータセットを処理します。PROC PRINT では 5 つのオブザベーションが印刷されます。これは WHERE 式に適合するすべてのオブザベーションで、5 番目の適合オブザベーションから開始されます。FIRSTOBS=オプションは、SPD Engine ではサポートされていません。

```
libname growth v9 'SAS-library';
data growth.teens;
  input Name $ Sex $ Age Height Weight;
  list;
datalines;
Alfred M 14 69.0 112.5
Carol F 14 62.8 102.5
James M 13 57.3 83.0
Janet F 15 62.5 112.5
Judy F 14 64.3 90.0
Philip M 16 72.0 150.0
Zeke M 14 71.1 105.1
Alice F 14 65.1 91.0
William M 15 66.5 112.0
Mike M 16 67.0 105.1
;
proc print data=growth.teens (firstobs=5);
  where age >13;
  title 'WHERE age>13 using the V9 Engine';
run;
```

**アウトプット 4.11** 印刷された 5 つのオブザベーション

WHERE age > 13 using the V9 Engine

| Obs | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 6 | Philip | M | 16 | 72.0 | 150.0 |
| 7 | Zeke | M | 14 | 71.1 | 105.1 |
| 8 | Alice | F | 14 | 65.1 | 91.0 |
| 9 | William | M | 15 | 66.5 | 112.0 |
| 10 | Mike | M | 16 | 67.0 | 105.1 |

## SYNCADD=データセットオプション

オブザベーションの追加時に、一度に 1 つのオブザベーションを処理するのか、それとも一度に複数のオブザベーションを処理するのかを指定します。

**該当要素:** PROC SQL

**デフォルト:** NO

**操作:** UNIQUESAVE=

**エンジン:** SPD Engine のみ

### 構文

**SYNCADD=**YES | NO

#### 必須引数

**YES**
一度に 1 つのオブザベーションを処理します(同期)。

**NO**
一度に複数のオブザベーションを処理します(非同期)。

### 詳細

SYNCADD=YES では、オブザベーションは一度に 1 つずつ処理されます。PROC SQL では、重複しないインデックスが付いたデータセットにオブザベーションを挿入した結果、SPD Engine で重複する値を含むオブザベーションが検出された場合、次の処理が発生します。

- 追加操作を停止
- 直前に追加したトランザクションをすべて取り消し
- ディスク上の元のデータセットは変更なし

SYNCADD=NO でオブザベーションを追加した方が、明らかに処理がずっと速くなります。ただし、PROC SQL を使用して、重複しないインデックスが付いたデータセットに少数のオブザベーションを挿入する場合、SPD Engine では、重複

値が 1 つ見つかると、すべてのオブザベーションが取り消されることがあります。具体的には、次の処理が発生します。

- SPD Engine でオブザベーションを拒否
- SPD Engine で処理を続行
- 追加操作または挿入操作の終了時にのみステータスコードを発行

拒否されたオブザベーションを別のデータセットに保存するには、UNIQUESAVE=データセットオプションを YES に設定します。

### 例: 重複しないインデックスが付いたデータセットに重複値のあるオブザベーションを挿入

次の例では、2 つの同一データセット WITH_NO および WITH_YES が作成されます。両方ともインデックスは重複していません。

PROC SQL を使用して、3 つの新しいオブザベーションが挿入されます。そのうちの 1 つには重複値があります。SYNCADD=YES オプションが使用されます。PROC SQL は重複値が検出されると停止し、データセットを復元します。

もう一度 PROC SQL を使用して、これら 3 つの新しいオブザベーションが挿入されます(前回と同様)。この場合は、SYNCADD=NO オプションが使用されます。重複値のあるオブザベーションは拒否されます。SAS ログは次のとおりです。

**ログ 4.5** オブザベーションの挿入

```
1 libname addlib spde 'c:\temp'; NOTE: Libref ADDLIB was successfully assigned as follows: Engine: SPDE Physical Name: c:\temp\ 2 3 data addlib.with_no(index=(x /unique)) 4 addlib.with_yes(index=(x /unique)) ; 5 input z $ 1-20 x y; 6 list; 7 datalines; RULE: ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+ 8 one 1 10 9 two 2 20 10 three 3 30 11 four 4 40 12 five 5 50 NOTE: The data set ADDLIB.WITH_NO has 5 observations and 3 variables.NOTE: The data set ADDLIB.WITH_YES has 5 observations and 3 variables.13 run; 14 15 proc sql; 16 insert into addlib.with_yes(syncadd=yes) 17 values('six_yes', 6, 60 ) 18 values('seven_yes', 2, 70 ) 19 values('eight_yes', 8, 80 ) 20 ; ERROR: Duplicate values not allowed on index x for file WITH_YES.NOTE: This insert failed while attempting to add data from VALUES clause 2 to the data set.NOTE: Deleting the successful inserts before error noted above to restore table to a consistent state.21 quit; NOTE: The SAS System stopped processing this step because of errors.22 23 proc sql; 24 insert into addlib.with_no(syncadd=no) 25 values('six_no', 6, 60 ) 26 values('seven_no', 2, 70 ) 27 values('eight_no', 8, 80 ) 28 ; NOTE: 3 rows were inserted into ADDLIB.WITH_NO.WARNING: Duplicate values not allowed on index x for file WITH_NO, 1 observations rejected.29 quit; 30 31 proc compare data=addlib.with_no compare=addlib.with_yes; 32 run; NOTE: There were 7 observations read from the data set ADDLIB.WITH_NO.NOTE: There were 5 observations read from the data set ADDLIB.WITH_YES.
```

## THREADNUM=データセットオプション

SPD Engine データセットの処理のために SPD Engine で起動可能な I/O スレッド最大数を指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**デフォルト:** SPDEMAXTHREADS=システムオプションの値(指定されている場合)。デフォルト値は、お使いのコンピュータの CPU 数を 2 倍した数

**操作:** “SPDEMAXTHREADS=システムオプション” (97 ページ)

**エンジン:** SPD Engine のみ

## 構文

**THREADNUM=*n***

### 必須引数

***n***
スレッド数を指定します。

## 詳細

THREADNUM=では、SPD Engine データセットの処理のために SPD Engine で起動される I/O スレッドの最大数を指定できます。THREADNUM=値は、次の SPD Engine I/O 処理のいずれにも適用されます。

- WHERE 式の処理
- 並列処理でのインデックス作成
- スレッド対応アプリケーションで要求される I/O

THREADNUM=の調整によって、システム管理者は、SPD Engine で任意のプロセスに使用可能な CPU リソースのレベルを調整できます。たとえば、64 ビットプロセッサシステムで THREADNUM=4 を設定すると、プロセスが多くとも 4CPU までに制限されるので、他のユーザーまたはアプリケーションのスループットを高められます。

THREADNUM=が 1 より大きい場合は、並列処理が発生する可能性が高くなります。したがって、物理的な順序が出力では保持されない場合があります。

また、このオプションを使用すると、WHERE 式評価のスケーラビリティも探索できます。

SPDEMAXTHREADS=システムオプションでは、システムリソースの消費に上限が課されるため、THREADNUM=値が制約されます。

*注:* SAS システムオプション NOTHREADS は SPD Engine には影響しません。

*注:* THREADNUM=1 を設定すると、並列処理は発生しなくなります。これはデフォルト Base SAS Engine と一致した動作です。

## 例: %MACRO の使用

SPD Engine システムオプション SPDEMAXTHREADS=がセッションに対して 128 に設定されます。次の例では、SAS マクロで並列処理の効果が示されます。

```
%macro dotest(maxthr);
%do nthr=1 %to &maxthr;
data _null_;
set spde cen.precs(threadnum= &nthr);
  where occup= '022'
  and state in('37','03','06','36');
run;
%mend dotest;
```

## UNIQUESAVE=データセットオプション

重複しないインデックスが付いたデータセットにオブザベーションを追加または挿入するときに、重複するキー値を含むオブザベーション(拒否されたオブザベーション)を別のデータセットに保存するように指定します。

**該当要素:** PROC APPEND および PROC SQL

**使用要素:** SPDSUSDS 自動マクロ変数

**デフォルト:** NO

**操作:** “SYNCADD=データセットオプション” (83 ページ)

**エンジン:** SPD Engine のみ

### 構文

**UNIQUESAVE=**YES | NO

#### 必須引数

**YES**
SYNCADD=NO の場合、拒否されたオブザベーションがシステム作成された別のデータセットに書き込まれます。このデータセットにアクセスするには、マクロ変数 SPDSUSDS を参照します。

**NO**
拒否されたオブザベーションは別のデータセットに書き込まれません。

### 詳細

インデックスが重複しないデータセットにオブザベーションを追加するときに、データセットオプション SYNCADD=NO が設定されている場合は、UNIQUESAVE=YES を使用します。

SYNCADD=NO では、追加操作または挿入操作で、オブザベーションを一度に 1 つずつではなくブロック単位で処理(パイプライン処理)するように指定されます。すべてのオブザベーションがデータセットに適用されて初めて、重複インデックス値が検出されます。UNIQUESAVE=YES の場合、拒否されたオブザベーションは別のデータセットに保存され、その名前が SPD Engine マクロ変数 SPDSUSDS に格納されます。データセット名のかわりにそのマクロ変数を指定して、拒否されたオブザベーションを識別できます。

*注:* SYNCADD=YES の場合、UNIQUESAVE=オプションは無視されます。詳細については、SYNCADD=データセットオプションを参照してください。

### 例: APPEND プロシジャでの UNIQUESAVE=オプションの使用

次の例では、2 つの重複しないインデックスが付いたデータセットが作成され、次に、重複値がある別のデータセットが最初のデータセットに追加されます。UNIQUESAVE=YES オプションが指定されているため、拒否されたオブザベーションを含むデータセットが作成されます。そのデータセットには、重複値があった変数を識別する変数が含まれます。SAS ログは次のとおりです。

**ログ 4.6** UNIQUESAVE=オプション

```
1    libname employee spde 'c:\temp'; NOTE: Libref EMPLOYEE was successfully assigned as follows:
Engine:        SPDE Physical Name: c:\temp\ 2 3    data employee.emp1 (index=(phone/unique room/
unique)); 4       input name $ phone room; 5       list; 6       datalines; RULE:      ----+----1----+----2----+----3----
+----4----+----5----+----6----+----7----+----8----+ 7          Jill 4344 456 8          Jack 5589 789 9          Jim  8888 345
10         Sam  3334 657 NOTE: The data set EMPLOYEE.EMP1 has 4 observations and 3 variables.11   run;
12 13   data employee.emp2; 14      input name $ phone room; 15      list; 16      datalines; RULE:      ----
+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+ 17         Jack  8443 679 18         Ann
3334 987 19         Sam   8756 346 20         Susan 5321 456 NOTE: The data set EMPLOYEE.EMP2 has 4
observations and 3 variables.21   run; 22 23   proc append base=employee.emp1(syncadd=no
uniquesave=yes) NOTE: Writing HTML Body file: sashtml.htm 24               data=employee.emp2; 25   run;
NOTE: Appending EMPLOYEE.EMP2 to EMPLOYEE.EMP1.NOTE: There were 4 observations read from the
data set EMPLOYEE.EMP2.NOTE: 2 observations added.NOTE: The data set EMPLOYEE.EMP1 has 6
observations and 3 variables.WARNING: Duplicate values not allowed on index phone for file EMP1, 1
observations rejected.WARNING: Duplicate values not allowed on index room for file EMP1, 1
observations rejected.NOTE: Duplicate records have been stored in file
EMPLOYEE._SPDEDUP048604700067A9F340C7E3E6.26 27   proc print data=employee.emp1; 28      title
'Listing of Final Data Set'; 29   run; NOTE: There were 6 observations read from the data set
EMPLOYEE.EMP1.30 31   proc print data=&spdsusds 32      title 'Listing of Rejected observations'; 33
run; NOTE: There were 2 observations read from the data set
EMPLOYEE._SPDEDUP048604700067A9F340C7E3E6.
```

**アウトプット 4.12** UNIQUESAVE=YES

**Listing of Final Data Set**

| Obs | name | phone | room |
|---|---|---|---|
| 1 | Jill | 4344 | 456 |
| 2 | Jack | 5589 | 789 |
| 3 | Jim | 8888 | 345 |
| 4 | Sam | 3334 | 657 |
| 5 | Jack | 8443 | 679 |
| 6 | Sam | 8756 | 346 |

**アウトプット 4.13** 拒否されたオブザベーション

**Listing of Rejected observations**

| Obs | name | phone | room | XXX00000 |
|---|---|---|---|---|
| 1 | Ann | 3334 | 987 | phone |
| 2 | Susan | 5321 | 456 | room |

## WHERENOINDEX=データセットオプション

WHERE 式の評価時に取り除くインデックスのリストを指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**デフォルト:** 空白

**制限事項:** IDXWHERE=NO データセットオプションと一緒には使用不可

**エンジン:** SPD Engine のみ

### 構文

**WHERENOINDEX=**(*name(s)*)

#### 必須引数

**(*name(s)*)**
WHERE プランナから取り除くインデックス名のリスト。

### 例: インデックスを取り除く

データセット PRECS がインデックスと一緒に定義されます。

```
proc datasets lib=spde cen
  modify precs;
  index create stser=(state serialno) occind=(occup industry) hour89;
quit;
```

次のクエリの評価時に、SPD Engine では、STATE 変数でも HOUR89 変数でもインデックスは使用されません。

この場合、OCCUP 変数と INDUSTRY 変数の条件の AND 組み合わせでは、出力対象は非常に少なくなります。条件を満たすオブザベーションは少数です。クエリの完全インデックス評価に必要とされる余分なインデックス I/O(コンピュータ時間)を回避するには、次の SAS コードを使用します。

```
proc sql;
  create data set hr80spde
  as select state, age, sex, hour89, industry, occup from spde cen.precs
    (wherenoindex=(stser hour89))
  where occup='022'
  and state in('37','03','06','36')
  and industry='012'
  and hour89 > 40;
 quit;
```

*注:* WHERENOINDEX リストでは、変数名ではなく、インデックス名を指定します。前述の例では、STATE 変数の複合インデックス STSER と単一インデックス HOUR89 の両方が取り除かれます。

5 章
# SPD Engine システムオプション



## SPD Engine システムオプションについて

SAS システムオプションは、SAS セッションに影響を与える指示です。SAS システムの初期化、ハードウェアやソフトウェアのインターフェイス、ジョブと SAS ファイルの入力、処理および出力などの、SAS が操作を実施する方法を制御します。SPD Engine システムオプションは SAS システムオプションと同じ方法で機能します。このセクションでは、SPD Engine でだけ使用されるシステムオプション、および SPD Engine では異なる動作をする Base SAS システムオプションについて説明します。

## 構文

OPTIONS *option-1 <option-2 ...>*;

*option*
変更する 1 つ以上の SPD Engine システムオプションを指定します。

次の例では、SPD Engine システムオプション MAXSEGRATIO=を指定します。

```
options maxsegratio=50;
```

*動作環境情報:* コマンドラインまたは構成ファイルでは、動作環境に固有の構文を使用します。詳細については、お使いの動作環境向けの SAS ドキュメントを参照してください。

## SPD Engine システムオプションのリスト

COMPRESS=
SPD Engine データセットの作成時にディスク上で圧縮するように指定します。

MAXSEGRATIO=
WHERE 式を処理する前に、候補セグメントとして識別するインデックスセグメントのパーセンテージを制御します。これは、インデックス付き変数を含む WHERE 式を評価するときに起こります。

MINPARTSIZE=
SPD Engine データセットの作成時に使用する最少パーティションサイズを指定します。

SPDFILECACHE | NOSPDEFILECACHE
オープンされている SPD Engine データファイルのキャッシングを有効化または無効化します。

SPDEINDEXSORTSIZE=
インデックスの作成で値を並べ替えるとき、並べ替えユーティリティが使用できるメモリ領域のサイズを指定します。

SPDEMAXTHREADS=
SPD Engine が I/O 処理のために起動できるスレッドの最大数を指定します。

SPDESORTSIZE=
SPD Engine によって使用される並べ替え処理操作に必要なメモリ領域サイズを指定します。

SPDEUTILLOC=
SPD Engine がユーティリティファイルを一時格納できる 1 つ以上のファイルシステムの場所を指定します。

SPDEWHEVAL=
どのオブザベーションが WHERE 式の条件を満たすかを判断するプロセスを指定します。

## SPD Engine とデフォルト Base SAS Engine で動作が異なる SAS システムオプション

MSGLEVEL=I
SAS ログに WHERE 最適化情報を出力します。

COMPRESS=
ユーザー定義の圧縮は実施できません。

DLDMGACTION=
DLDMGACTION=NOINDEX はサポートしていませんが、ABORT、 FAIL、PROMPT、REPAIR はサポートしています。

DLCREATEDIR
SPD Engine では機能しません。

ERRORS=MAX
オブザベーションの最大数を 2147483647 として、SAS はエラーメッセージを発行します。

FIRSTOBS=
SPD Engine では使用できません。

SORTPGM=
BEST オプションを使用した場合にパフォーマンス上の問題が発生する可能性があります。

VALIDMEMNAME=
VALIDMEMNAME=EXTEND を使用する場合は、メンバ名に次の制約があります。

- *class.group* など、ピリオド付きのメンバ名は使用不可
- *$class* など、$から始まるメンバ名は使用不可

VALIDVARNAME=
変数名に次の特殊文字のいずれかが含まれている場合は、変数に対して単一インデックスも複合インデックスも作成できません。

" * | \ : / < > ? -

# ディクショナリ

## COMPRESS=システムオプション

SPD Engine データセットの作成時にディスク上で圧縮するように指定します。

**該当要素:** 構成ファイル、SAS 起動時、OPTIONS ステートメント、**システムオプション**ウィンドウ

**カテゴリ:** システム管理: パフォーマンス

**デフォルト:** NO

**制限事項:** ENCRYPT=YES または ENCRYPT=AES と一緒には使用不可

**操作:** “IOBLOCKSIZE=データセットオプション” (73 ページ)
“PADCOMPRESS=データセットオプション” (77 ページ)

**エンジン:** SPD Engine のみ

### 構文

**COMPRESS=**NO | YES | CHAR | BINARY

### 必須引数

**NO**
データセットの圧縮は実行しません。

**YES | CHAR**
RLE (ランレングスエンコーディング)を使用して SPD Engine データセット内のデータをブロック単位で圧縮するように指定します。RLE では、反復実行される同じ文字(空白を含む)を 2 バイトまたは 3 バイトの表現に削減することでデータが圧縮されます。

**BINARY**
RDC (ロスデータ圧縮)を使用して SPD Engine データセット内のデータをブロック単位で圧縮するように指定します。RDC では、RLE とスライディングウィンドウ圧縮を組み合わせて反復バイトパターンをより効果的に表現することでファイルが圧縮されます。

*注:* これは、中容量から大容量(数百バイト以上)のブロックのバイナリデータ(文字変数と数値変数)を圧縮する場合に大きな効果を発揮する方法です。

## 詳細

COMPRESS=YES|BINARY|CHAR を指定した場合、SPD Engine では、データコンポーネントファイルが、作成時にブロックで圧縮されます。圧縮ブロックのサイズを指定するには、データセットの作成時に“IOBLOCKSIZE=データセットオプション” (73 ページ)を使用します。新しく圧縮したブロックに埋め込みを追加するには、データセットの作成または更新時に“PADCOMPRESS=データセットオプション” (77 ページ)を指定します。詳細については、“SPD Engine データセットの圧縮” (19 ページ)を参照してください。

SPD Engine では、ユーザー指定の圧縮はサポートされません。圧縮と暗号化の両方が行われたデフォルト Base SAS Engine データセットを移行する場合、暗号化は保持されますが、圧縮は解除されます。

CONTENTS プロシジャでは、圧縮設定が識別されます。データセットが圧縮される場合、PROC CONTENTS で圧縮についての情報が出力されます。次の例では、CONTENTS プロシジャ出力の圧縮情報フィールドについて説明します。

通常、COMPRESS=CHAR を指定すると、単一バイトの反復が存在する場合に適切な圧縮結果がもたらされ、COMPRESS=BINARY を指定すると、バイト列の反復が存在する場合に適切な圧縮結果がもたらされます。また、単一バイトの反復よりも、バイト列の反復を見つける方が労力を要します。たとえば、“例 1: COMPRESS=CHAR” (61 ページ)および“例 2: COMPRESS=BINARY” (62 ページ)を参照してください。

**アウトプット 5.1** PROC CONTENTS 圧縮情報セクション

| - Compressed Info | - |
|---|---|
| Number of compressed blocks | 202 |
| Raw data blocksize | 32736 |
| Number of blocks with overflow | 5 |
| Max overflow chain length | 3 |
| Block number for max chain | 80 |
| Min overflow area | 87 |
| Max overflow area | 181 |

圧縮ブロック数
　データの格納に必要な圧縮ブロック数。

生データブロックサイズ
　IOBLOCKSIZE=データセットオプションに指定したサイズから計算されたバイト単位の圧縮ブロックサイズ。これは、ブロックサイズ単位で取得するオブザベーション長の最大倍数です。

オーバーフローのあるブロック数
　より多くの領域を必要とする圧縮ブロック数。データが更新されて、圧縮された新ブロックが圧縮された旧ブロックよりも大きくなった場合、オーバーフローブロックフラグメントが作成されます。

オーバーフローチェーンの最大長
　単一ブロックのオーバーフロー最大数。たとえば、圧縮ブロックが更新されて大きくなり、再度更新されてより大きなサイズになった場合、オーバーフローチェーンの最大長は 2 になります。

最大チェーンのブロック数
　最大数のオーバーフローブロックを含むブロック数。

最小オーバーフロー領域
　オーバーフローに必要なディスク領域最小容量。

最大オーバーフロー領域
　オーバーフローに必要なディスク領域最大容量。

通常、圧縮ファイルへのアクセスには、より多くの処理時間を必要とします。読み取る前にファイルの圧縮解除が必要です。また、更新の場合は、ディスクに書き込む際に、もう一度圧縮する必要があります。

### 比較

COMPRESS=システムオプションは、COMPRESS= LIBNAME ステートメントオプションおよび COMPRESS=データセットオプションによって上書きされます。

## MAXSEGRATIO=システムオプション

WHERE 式を処理する前に、候補セグメントとして識別されるインデックスセグメントのパーセンテージを制御します。これは、インデックス付き変数を含む WHERE 式を評価するときに起こります。

**該当要素:** 構成ファイル、SAS 起動時、OPTIONS ステートメント、**システムオプション**ウィンドウ

**カテゴリ:** システム管理: パフォーマンス

**デフォルト:** 75

**エンジン:** SPD Engine のみ

## 構文

**MAXSEGRATIO=*n***

### 必須引数

***n***

SPD Engine が、WHERE 式で参照される値を含むと識別するインデックスセグメントのパーセンテージの上限を指定します。デフォルトは 75 で、SPD Engine が次を実施することを指定します。

- インデックスを使用して、特定の WHERE 式の値を含んでいるセグメントを識別します。
- 全セグメントの 75%より多くがその値を含むことが判明した場合、候補セグメントの識別を停止します。

有効な値の範囲は 0 と 100 の間の整数です。n=0 の場合、SPD Engine は候補セグメントを識別しようとせず、かわりに、すべてのセグメントへ WHERE 式を適用します。n=100 の場合は、SPD Engine が 100%のセグメントをチェックして候補セグメントを識別し、次に、それらの候補セグメントにだけ WHERE 式を適用します。

## 詳細

インデックス付き変数での WHERE 照会では、SPD Engine は、WHERE 式の 1 つ以上の条件と一致する 1 つ以上の変数値を含むインデックスセグメントの数を判別します。WHERE 式がその WHERE 式を満たすオブザベーションを含んでいるセグメントにのみ適用される場合、実質的パフォーマンスの向上が実現する場合がしばしばあります。

SPD Engine は、MAXSEGRATIO=の値を使用して、すべてのセグメントに WHERE 式を適用するコストが、候補セグメントを識別し続けるコストより少ない点がどこかを決定します。計算された比率が MAXSEGRATIO=で指定した比率を超過したとき、SPD Engine は候補セグメントの識別を停止し、すべてのセグメントへ WHERE 式を適用します。

*注:* テーブルによっては、75%が最適な設定にならない場合もあります。よりよい設定を決定するためには、パフォーマンスベンチマークを実行してパーセンテージを調整し、再度パフォーマンスベンチマークを実行してください。結果を比較することで、照会している特定のデータ母集団がインデックスセグメント比率の変化にどのように応答するかがわかります。

## 例

### 例 1: インデックスセグメントの識別

次の例では、SPD Engine は WHERE 式を満たすインデックスセグメントの識別を開始し、セグメントの総数と比較して、識別されたセグメントのパーセンテー

ジが 65 を超過するまで実施します。パーセンテージが 65 を超えた場合、SPD Engine は候補セグメントの識別を停止し、すべてのセグメントへ WHERE 式を適用します。

```
options maxsegratio=65;
```

### 例 2: WHERE 式の全セグメントへの適用

次の例では、SPD Engine がいずれの候補セグメントも最初から識別せず、WHERE 式を全セグメントへ適用します。

```
options maxsegratio=0;
```

### 例 3: 全インデックスセグメントの無停止評価

次の例では、SPD Engine がインデックスセグメントの識別を開始し、全セグメントが評価されるまで停止しません。その後で、識別されたすべての候補セグメントに WHERE 式が適用されます:。

```
options maxsegratio=100;
```

## MINPARTSIZE=システムオプション

SPD Engine データセットの作成時に使用する最小パーティションサイズを指定します。

**該当要素:** 構成ファイル、SAS 起動時

**カテゴリ:** システム管理: パフォーマンス

**デフォルト:** 16M

**エンジン:** SPD Engine のみ

### 構文

**MINPARTSIZE=***n* | *n*K | *n*M | *n*G

#### 必須引数

***n***

パーティションのサイズは、バイト、KB、MB、GB のいずれかです。最小パーティションサイズの最大値は 2GB–1、すなわち 2047MB です。

制限事項 この制限は、オペレーティングシステム z/OS、Linux SLES 9 x86 および Windows ファミリの 32 ビットホストにのみ適用されます。パーティションサイズが 2GB 以上のデータセットを作成した場合、SAS 9.2 以前のバージョンの SPD Engine では、そのデータセットを開くことはできません。次のエラーメッセージが SAS ログに書き込まれます。
**ERROR: Unable to open data file because its data representation differs from the SAS session data representation.**

### 詳細

MINPARTSIZE=の指定によって、PARTSIZE=オプションで指定したパーティションサイズに下限を設定することができます。MINPARTSIZE=の指定は、パーティションがほぼ同数のオブザベーションで作成されるかどうかに影響する場合があります。パーティションサイズが小さい場合は、処理時により多くのファイル

がオープンされることを意味します。使用するオペレーティングシステムによっては、使用ファイルのオープン数が制限される可能性があります。

## SPDEFILECACHE システムオプション

オープンされている SPD Engine データファイルのキャッシングを有効化または無効化します。

**該当要素:** 構成ファイル、SAS 起動時

**カテゴリ:** システム管理: パフォーマンス

**デフォルト:** NOSPDEFILECACHE

**エンジン:** SPD Engine のみ

### 構文

**SPDEFILECACHE | NOSPDEFILECACHE**

#### アクション

**SPDEFILECACHE**
オープンされている SPD Engine データファイルのキャッシングを有効化します。

**NOSPDEFILECACHE**
オープンされている SPD Engine データファイルのキャッシングを無効化します。

### 詳細

SPDEFILECACHE システムオプションを使用すると、オープンされている SPD Engine データファイルのキャッシングを有効化または無効化できます。SPD Engine データファイルをキャッシングすると、ファイルのオープンおよびクローズの必要性を減らすことにより、パフォーマンスを改善できます。SPD Engine データファイルを共有環境で使用している場合、ファイルをキャッシングするとパフォーマンスが低下します。このようなパフォーマンスの低下は、あるユーザーが、別のユーザーがオープンしたファイル(ただし、同ユーザーはもはやそのファイルにアクセスしていない)にアクセスする際に、そのファイルがクローズされるまで待機しなければならない場合に発生します。

## SPDEINDEXSORTSIZE=システムオプション

インデックスの作成で値を並べ替えるとき、並べ替えユーティリティが使用できるメモリ領域のサイズを指定します。

**該当要素:** 構成ファイル、SAS 起動時、OPTIONS ステートメント、**システムオプション**ウィンドウ

**カテゴリ:** システム管理: パフォーマンス

**デフォルト:** 32M

**エンジン:** SPD Engine のみ

## 構文

**SPDEINDEXSORTSIZE=***n* | *n*K | *n*M | *n*G

### 必須引数

***n***
バイト、KB、MB、GB のいずれかでのメモリ領域のサイズです。*n*=0 のとき、並べ替えユーティリティではデフォルトが使用されます。有効な値の範囲は、1,048,576 から 10,736,369,664 バイトまでです。

## 詳細

SPD Engine では、1 回のデータセットスキャンで複数のインデックスを作成できます。SPD Engine では、作成されたインデックスごとに 1 つずつスレッドが起動され、そのスレッドが同時に処理されます。各インデックスのデータセットをスキャンするよりも、インデックスを並列作成する方がずっと迅速に処理できますが、このオプションのデフォルトは NO です。これは、並列インデックス作成では、並べ替えファイルを格納するための追加ユーティリティ領域、および追加メモリが必要になるためです。リソース不十分のためインデックス作成が失敗した場合は、次のどちらかまたは両方を実行します。

- SPDEUTILLOC=システムオプションを使用して、ユーティリティファイル領域のサイズを増やします。
- SAS システムオプションを MEMSIZE=0 に設定し、[1]SPDEINDEXSORTSIZE=システムオプションを使用して、インデックスの並べ替えに使用されるユーティリティ領域を増やします。

最大の SPDEINDEXSORTSIZE=値は 10 GB ですが、2GB までに制限されているホストシステムではこの値は尊重されません。64 ビットの LONG データタイプを持つホストシステムでは、SPD Engine は 2GB より大きな値を有効とします。32 ビットの LONG データタイプを持つホストシステムでは、SPD Engine はメモリのみ 2GB まで使用できます。SPDEINDEXSORTSIZE オプションの値をより大きな値に設定できますが、より大きな値は有効ではありません。

*注:* より大きな値が有効ではないのに使用されている場合には、SAS ログに警告を受け取ります。

# SPDEMAXTHREADS=システムオプション

SPD Engine が I/O 処理のために起動できるスレッドの最大数を指定します。

**該当要素:** 構成ファイル、SAS 起動時
**カテゴリ:** システム管理: パフォーマンス
**デフォルト:** 0
**エンジン:** SPD Engine のみ

## 構文

**SPDEMAXTHREADS=***n*

[1] z/OS の場合、REGION サイズを増やします。

### 必須引数

***n***

SPD Engine が起動できるスレッドの最大数です。有効な値の範囲は、0 から 65,536 です。デフォルトサイズは 0 で、THREADNUM=の値が設定されている場合には SPD Engine はそれを使用します。設定されていない場合、SPD Engine は起動するスレッドの数を CPU 数に相当する数に設定します。

### 詳細

SPDEMAXTHREADS=の指定によって、SPD Engine 処理のために起動するスレッドの数に上限を設定します。それには次が含まれます。

- WHERE 式の処理
- 並列処理でのインデックス作成
- SAS スレッド対応プロシジャなどのスレッド対応アプリケーションによって要求された任意の I/O 処理

SPDEMAXTHREADS=は、THREADNUM=データセットオプションを抑制します。

## SPDESORTSIZE=システムオプション

SPD Engine によって使用される並べ替え処理操作に必要なメモリ領域サイズを指定します。

**該当要素:** 構成ファイル、SAS 起動時、OPTIONS ステートメント、**システムオプション**ウィンドウ

**カテゴリ:** システム管理: パフォーマンス

**デフォルト:** 32M

**エンジン:** SPD Engine のみ

### 構文

**SPDESORTSIZE=***n* | *n*K | *n*M | *n*G

### 必須引数

***n***

バイト、KB、MB、GB のいずれかでのメモリ領域のサイズです。*n*=0 のとき、並べ替えユーティリティではデフォルトが使用されます。有効な値の範囲は、1,048,576 から 10,736,369,664 バイトまでです。

### 詳細

SPD Engine は、自動並べ替えを並列処理で実行できます。SPDESORTSIZE=に指定する並べ替えサイズは、並列プロセス数の倍数にする必要があります。この並べ替えサイズの合計は、プロセスで使用できる物理メモリより小さくする必要があります。SPDESORTSIZE=の適切な指定により、動作環境で制御されるメモリのスワップが抑えられ、パフォーマンスが改善される場合があります。

並べ替え処理が指定したよりも多くのメモリを必要とする場合は、次のうちの一つを実施してください。

- SAS システムオプション MEMSIZE=0 で SAS を再起動します(z/OS の場合、REGION サイズを増やします)。

- SPDEUTILLOC=システムオプションを使用して、ユーティリティファイル領域のサイズを増やします。

SPDEINDEXSORTSIZE=オプションを使用して、インデックスの作成で値を並べ替えるときに使用されるメモリを増やします。インデックスの並列作成を指定する場合は、SPDEUTILLOC=システムオプションを使用して十分な容量の領域を指定します。

*注:* デフォルト Base SAS Engine 用ドキュメントに説明されている SORTSIZE=オプションは、PROC SORT 操作に影響します。SPDESORTSIZE=の指定は SPD Engine に特有の並べ替え操作に影響を与えます。

最大の SPDESORTSIZE=値は 10 GB ですが、2GB までに制限されているホストシステムではこの値は尊重されません。64 ビットの LONG データタイプを持つホストシステムでは、SPD Engine は 2GB より大きな値を有効とします。32 ビットの LONG データタイプを持つホストシステムでは、SPD Engine はメモリのみ 2GB まで使用できます。SPDESORTSIZE オプションの値をより大きな値に設定できますが、より大きな値は有効ではありません。

*注:* より大きな値が有効ではないのに使用されている場合には、SAS ログに警告を受け取ります。

## SPDEUTILLOC=システムオプション

SPD Engine がユーティリティファイルを一時格納できる 1 つ以上のファイルシステムの場所を指定します。

**該当要素:** 構成ファイル、SAS 起動時

**カテゴリ:** システム管理: パフォーマンス

**エンジン:** SPD Engine のみ

**参照項目:** システムオプションの指定方法は、使用する動作環境に合った SAS ドキュメントに説明されています。

### 構文

**-SPDEUTILLOC** *directory | ("directory-1" "directory-2"...)*

#### 必須引数

***location***
ユーティリティファイルが作成される既存のディレクトリです。

***("directory-1" "directory-2" ...)***
ユーティリティファイルが作成される一連の既存のディレクトリです。一重引用符または二重引用符を使用できます。

注 *location* は、一重引用符または二重引用符で囲むことができます。*location* にブランクが埋め込まれている場合は引用符で囲む必要があります。

### 詳細

*動作環境の情報*

システムオプションの指定方法は、使用するオペレーティングシステム用 SAS ドキュメントに説明されています。

SPD Engine は、自動並べ替えやインデックス作成などの特定の処理中に一時ユーティリティファイルを作成します。処理を正常に完了させるには、ユーティリティファイルを格納するために十分な領域が必要です。SPDEUTILLOC=システムオプションを使用して、処理のために十分な適当量の領域を指定できます。ただし、HP Integrity サーバーの OpenVMS では、ライブラリは ODS-5 ファイルである必要があります。SPDEUTILLOC=システムオプションに複数のディレクトリが指定されている場合、最初のユーティリティファイルのディレクトリは処理の開始時にランダムに選択されます。周期的な方法で他のディレクトリへ選択が続きます。ユーティリティファイルは一時的で、処理が完了した後に削除されます。

*注:* 構文エラーを回避するためには、構成ファイルで複数のディレクトリを指定してください。

ユーティリティファイルの作成処理に十分な領域を確保するため、SPDEUTILLOC=オプションあるいは UTILLOC=オプションを常に指定することをお勧めます。次のような場合には、一時ユーティリティファイル格納のための場所は動作環境ごとに定義されます。

- SPDEUTILLOC=システムオプションまたは UTILLOC= SAS システムオプションが指定されていない。
- SPD Engine が SAS WORK ディレクトリを検出できない(あるいはそれに対する書き込み許可がない)

次の表はデフォルトのユーティリティファイルの場所を示します。

**表 5.1** ユーティリティファイルのデフォルト場所

| 動作環境 | デフォルト場所 1 | デフォルト場所 2 | デフォルト場所 3 |
|---|---|---|---|
| | | | |
| UNIX | UTILLOC= SAS システムオプション (指定がある場合) | SAS ワークライブラリ | /tmp |
| Windows | UTILLOC= SAS システムオプション (指定がある場合) | SAS ワークライブラリ | TEMP=環境変数で指定された場所 |
| z/OS | UTILLOC= SAS システムオプション (指定がある場合) | SAS ワークライブラリ | /tmp |

## SPDEWHEVAL=システムオプション

どのオブザベーションが WHERE 式の条件を満たすかを判断するプロセスを指定します。

**該当要素:** 構成ファイル、SAS 起動時、OPTIONS ステートメント、**システムオプション**ウィンドウ

**カテゴリ:** システム管理: パフォーマンス

**デフォルト:** COST

**エンジン:** SPD Engine のみ

## 構文

**SPDEWHEVAL=**COST | EVAL1 | EVAL3EVAL4

### 必須引数

**COST**
WHERE 式を最適化するために使用する評価方法を SPD Engine が決定することを指定します。この処理では、使用するスレッド数を計算します。これにより、十分使用されなかったスレッドを起動するオーバーヘッドが最小化されます。これがデフォルト設定です。

**EVAL1**
マルチスレッドのインデックス評価方法であり、複数のスレッドを使用することで、WHERE 式を満たす行を迅速に判断できます。オブザベーションを取得するために起動されるスレッドの数は、THREADNUM=の値と同じです。

**EVAL3EVAL4**
単一スレッドのインデックス評価方法であり、単純あるいは複合の WHERE 式に使用されます。すべてのキー変数は単純なインデックスを持ち、非同等性を検査する条件は持ちません。オブザベーションの取得にはマルチスレッドが使用される場合もあります。

## 詳細

SPD Engine は、WHINIT (規則に基づく WHERE 式プランナ)を使用して、最もクエリに適した評価方法を選択します。SAS システムオプション MSGLEVEL=I によって、WHINIT メッセージが SAS ログに出力されます。これは、1 つ以上のインデックスがクエリで使用されるかどうかを決定するのに役立ちます。

COST は SPDEWHEVAL=のデフォルト設定で、WHERE 式および任意の使用可能なインデックスを分析します。分析に基づき、SPD Engine が WHERE 式を最適化する評価方法を選択します。評価方法は、EVAL1、EVAL3、EVAL4 のいずれかになります。インデックスが利用可能でない場合、データを順次読み込む方法を選択できます。または、分析の結果、インデックスの使用では処理時間を改善できないことが示されている場合にも、データを順次読み込む方法を選択できます。

COST は、WHERE 式の処理に使用するスレッドの数を最適化します。COST は、効率的に使用できるスレッドの数を決定して起動します。THREADNUM=の値に基づき、COST は使用されていないスレッドを起動しないことで処理時間を大幅に節約できます。

その WHERE 式が他の評価方法基準に厳密に適合する場合以外は、COST が SPDEWHEVAL=のための推奨値です。COST 以外の値がより効率的かどうかを判断するためには、ベンチマークテストの使用を強くお勧めします。

EVAL1 は、WHERE 式が複雑で、変数に複数のインデックスがあるときにはより効率的な場合があります。EVAL1 では、複数のスレッドを起動して、どのセグメントが WHERE 式の条件を満たすかを判断します。また、オブザベーションの取得にも複数のスレッドを使用できます。

*注:* COST が最適でない場合も多少あります。EVAL1 または EVAL3EVAL4 に値を変更することでよりよいパフォーマンスを産めるかどうかは、パフォーマンスベンチマークを実行し、値を変更して再度パフォーマンスベンチマークを実行することで判断できます。結果を比較することで、照会している特定

のデータ母集団が、計画中の規則に基づいた WHERE にどのように応答するかがわかります。

# 推奨資料

このタイトルに関連した推奨される参考資料のリストを次に示します。

- *Base SAS Procedures Guide*
- *SAS Data Set Options: Reference*
- *SAS Language Reference: Concepts*
- *SAS System Options: Reference*
- *The Little SAS Book: A Primer*

SAS 刊行物の一覧については、 sas.com/store/books から入手できます。必要な書籍についての質問は SAS 担当者までお寄せください:

SAS Books
SAS Campus Drive
Cary, NC 27513-2414
電話: 1-800-727-0025
ファクシミリ: 1-919-677-4444
メール: sasbook@sas.com
Web アドレス: sas.com/store/books

# 用語集

**CPU バウンドアプリケーション**
パフォーマンスがデータ計算の実行可能速度に制約されるアプリケーション。複数の CPU とスレッドテクノロジによってこの問題を軽減できます。

**I/O バウンドアプリケーション**
パフォーマンスが処理データの配信可能速度に制約されるアプリケーション。複数の CPU、分割 I/O、スレッドテクノロジ、RAID (redundant array of independent disks)テクノロジ、またはこれらの組み合わせによって、この問題を軽減できます。

**RAID**
*参照項目:* redundant array of independent disks.

**redundant array of independent disks (RAID)**
安価で大容量のデータを記憶でき多数のディスクで構成されるインタリーブ記憶システムの一種。RAID では複数のレベルを使用できます。たとえば、レベル 0 の RAID では、2 つ以上のハードドライブが結合されて 1 つの論理ディスクドライブになります。さまざまな RAID レベルで、それぞれ異なる冗長性と記憶容量が提供されます。また、同じデータが異なる場所に格納されるため、I/O 操作を同時進行させられます。その結果、パフォーマンスが向上する場合があります。 *関連項目:* 冗長性.

**SAS Scalable Performance Data Engine: (SPD Engine)**
データを効率化されたファイル形式に編成し、データのアプリケーションへの迅速な配信を可能にする SAS エンジン。 *関連項目:* 並列 I/O, 並列処理.

**sasroot**
サイトまたはコンピュータで SAS がインストールされているディレクトリまたはフォルダの名前を示す表記。

**SMP**
*参照項目:* 対称型マルチプロセッシング.

**SPD Engine**
*参照項目:* SAS Scalable Performance Data Engine:.

**SPD Engine データセット**
SPD Engine で作成されるデータセットで、コンポーネントファイルが 4 つまで含まれます(データ用が 1 つ、メタデータ用が 1 つ、任意のインデックス用

が 2 つ)。最小コンポーネントファイル数は、データとメタデータの 2 つです。SPD Engine のファイル編成では、データとメタデータは分けられます。

**SPD Engine データファイル**
SPD Engine データセットのデータコンポーネント。SAS データファイルとは対照的に、SPD Engine データファイルに含まれるのはデータのみです。メタデータは含まれません。SPD Engine では、データビューはサポートされません。 *関連項目:* SPD Engine データセット.

**WHERE 式**
オブザベーションの選択基準を定義する、WHERE 句内の構文文字列。たとえば、会員データベースでは、式"WHERE member_type=Senior"によって、すべてのシニア会員が返されます。 *関連項目:* 複合 WHERE 式.

**解決時間**
タスクの完了に必要な経過時間。解決時間の測定値は、異なるコンピューティング環境におけるソフトウェアアプリケーションのパフォーマンス比較に使用されます。言い換えれば、スケーラビリティの測定に使用されます。 *関連項目:* スケーラビリティ.

**起動**
プロセスまたはプロセススレッド(軽量プロセススレッド(LWPT)など)を開始すること。 *関連項目:* スレッド.

**軽量プロセススレッド**
通常はオペレーティングシステムコールによって個別に作成および制御される単一スレッドサブプロセス。複数の軽量プロセススレッドは、対称型マルチプロセッシング(SMP)ハードウェアかまたはスレッド対応オペレーティングシステムで一度にアクティブにできます。

**コントローラ**
コンピュータと、ディスクや RAID などの周辺デバイスとの間の相互作用を管理するコンピュータコンポーネント。たとえば、コントローラでは、CPU とディスクドライブとの間のデータ I/O が管理されます。コンピュータには多くのコントローラを含められます。1 つの CPU で 2 つ以上のコントローラに命令できます。また、1 つのコントローラで複数のディスクに命令できます。

**サーバースケーラビリティ**
複数のクライアント要求を同時に処理するために SMP ハードウェアとスレッド処理を利用するサーバー機能。したがって、SMP ハードウェアで提供される計算能力が増加すると、それに比例して時間単位ごとの処理可能トランザクション数が増加します。 *関連項目:* 対称型マルチプロセッシング.

**冗長性**
障害またはエラー、あるいはその両方の影響を最小限に抑えるために、複数の交換可能な構成要素が提供されるというコンピューティングシステムの特性。たとえば、データが(RAID などで)冗長的に格納されている場合、1 つのディスクが失われても、そのデータは別のディスクではなおも使用可能です。 *関連項目:* redundant array of independent disks.

**スケーラビリティ**
コンピューティング環境の変化や、計算、ユーザー、またはデータの量の変化にかかわらず、ソフトウェアアプリケーション十分に機能させてパフォーマンスの損失を最小限に抑える機能。スケーラブルなソフトウェアでは、SMP ハードウェアおよびスレッド処理の使用によってもたらされるような計

算機能の増大を最大限に活用できます。 *関連項目:* スケーラブルなソフトウェア.

**スケーラブルなソフトウェア**
SMP ハードウェアの計算機能の増大に予測どおりに対応するソフトウェア。たとえば、CPU 数を増加した場合、CPU バウンドの問題解決までの時間は、それに比例して減少します。また、I/O システムのスループットが増加した場合、I/O バウンドの問題解決までの時間は、それに比例して減少します。

**スレッド**
オペレーティングシステムによりスケジューリング可能な処理の最小単位。

**スレッド I/O**
速度増加のために複数のスレッドによって実行される I/O。スレッド I/O でパフォーマンスを大幅に向上させるには、I/O を実行しているアプリケーションにもデータの迅速な処理能力が必要です。 *関連項目:* I/O バウンドアプリケーション, スレッド.

**スレッド処理**
CPU バウンドアプリケーションの速度を向上させるために複数のスレッドで実行される処理。 *関連項目:* CPU バウンドアプリケーション, 対称型マルチプロセッシング.

**スレッド処理**
データ処理かまたはデータ I/O のハイパフォーマンステクノロジです。この場合、タスクがスレッドに分割され、1 つ以上の CPU の複数のコアで同時実行されます。

**スレッド対応オペレーティングシステム**
複数の CPU による共有メインメモリ領域への対称アクセスの調整が可能なオペレーティングシステム。この調整アクセスによって、同じプロセスのスレッドが非常に効率的にデータを共有できるようになります。

**スレッド対応プロシジャ**
スレッド I/O またはスレッド処理をサポートする SAS プロシジャ。

**ソートインジケータ**
データセットを並べ替えるかどうか、どのように並べ替えたか、および並べ替えが検証されたかどうかを示すデータファイルの属性。ソートインジケータは特に次の情報を示します。1) 並べ替えに使用された BY 変数。2) 文字変数に使用された文字セット。3) 使用された文字変数の照合順序。4) 並べ替え情報が検証されたかどうか。この属性は、データファイルディスクリプタ情報に格納されます。プロセスの一部としてデータの並べ替えを必要とする SAS プロシジャはすべて、ソートインジケータを使用します。

**対称型マルチプロセッシング (SMP)**
I/O 速度と処理速度の向上を可能にするハードウェアとソフトウェアのアーキテクチャの一種。SMP マシンには、複数の CPU とスレッド対応オペレーティングシステムが搭載されています。SMP マシンは通常、複数のコントローラ、およびコントローラごとの複数のディスクドライブで構成されます。

**データパーティション**
データが含まれる物理ファイルで、SAS Scalable Performance Data Engine データセットのデータコンポーネントから成る物理ファイルのコレクションの一部。 *関連項目:* パーティション.

**パーティション**
複数のデバイスやディレクトリにわたる論理ファイルの一部または全部。SPD Engine では、パーティションは 1 つの物理ファイルです。データファイル、インデックスファイル、メタデータファイルはすべて分割可能で、その結果、それぞれデータパーティション、インデックスパーティション、メタデータパーティションが生成されます。ファイルの分割によって、非常に大きなデータセットでパフォーマンスの向上が可能になります。 *関連項目:* データパーティション.

**複合 WHERE 式**
WHERE X=1 と Y>3 などのように、2 つ以上の演算子を含む WHERE 式。 *関連項目:* WHERE 式.

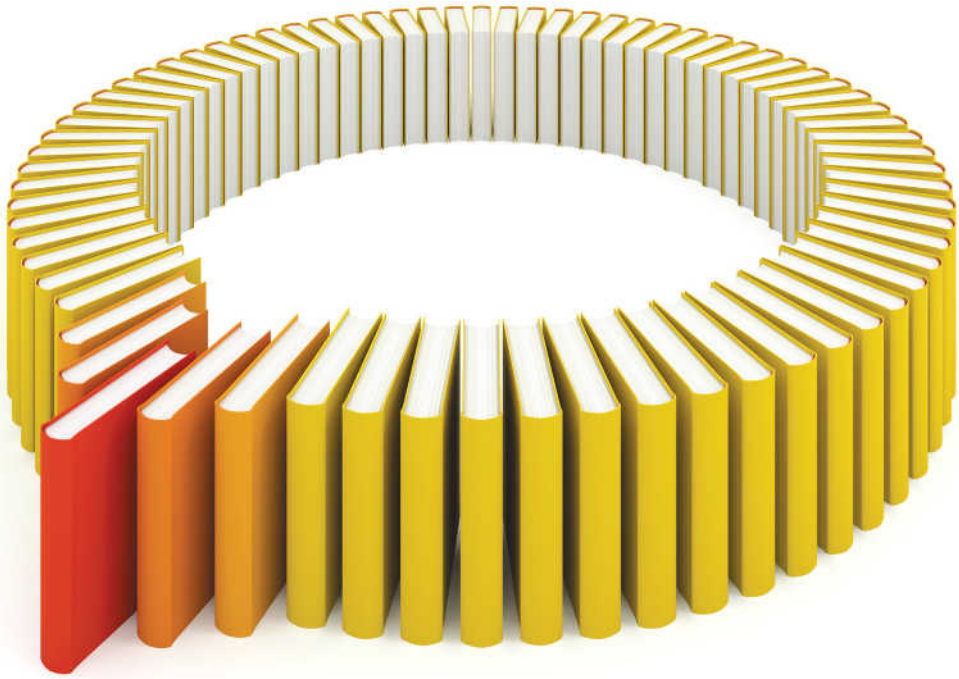
**プライマリパス**
SPD Engine メタデータファイルが格納される場所。複数 CPU のパフォーマンス向上を活用するために、その他の SPD Engine コンポーネントファイル(データファイルとインデックスファイル)は別の記憶域パスに格納されます。

**ブロック**
データセット内のオブザベーションのグループ。ブロックを使用すると、スレッド対応アプリケーションで、オブザベーションの読み取り、書き込みおよび処理が、個別オブザベーションとして実施される場合よりも迅速に行えます。

**並列 I/O**
複数の CPU と複数のコントローラを利用した入出力方法で、コントローラごとに複数のディスクを使用して、独立したスレッドでデータの読み取りや書き込みを行います。

**並列処理**
大きなジョブを分割し、複数の CPU で同時実行可能な複数の小さなジョブにする処理方法。 *関連項目:* スレッド処理.

# キーワード



# Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

support.sas.com/bookstore
*for additional books and resources.*